Rapport de stage - M2 MPRI

Décodage Statistique

Thomas Debris

12/08/2016

Décodage Statistique

Thomas Debris, Jean-Pierre Tillich, Equipe SECRET, Inria Paris

03/2016-09/2016

Le contexte général

Depuis la naissance de la cryptographie à clef publique, les systèmes de chiffrement utilisés dans la pratique reposent sur la seule famille des problèmes issus de la théorie des nombres. La cryptographie asymétrique est aujourd'hui complètement dépendante de deux uniques problèmes -qui ne sont d'ailleurs pas si éloignés-, celui du logarithme discret et celui de la factorisation. De nombreux efforts ont dès lors été fait par la communauté scientifique afin, d'une part prouver la sécurité de ces systèmes et, de l'autre, chercher à les casser. Il s'avère que la seule réponse apportée fut d'augmenter encore et encore la taille des clés face à des attaques de plus en plus efficaces. L'hypothèse de nouvelles failles de RSA semble même encore plus menaçante car il n'a jamais été prouvé qu'il était nécessaire de savoir factoriser une clef publique pour déchiffrer. Le problème du logarithme discret a quant à lui été partiellement cassé dans [1] en 2013. Barbulescu, Gaudry, Joux et Thomé, ont en effet trouvé un algorithme quasi-polynomial ($n^{O(\log(n))}$) résolvant ce problème pour des petites caractéristiques. Le danger s'approchant, la recherche et l'étude de nouveaux paradigmes sur lesquels fonder la cryptographie asymétrique est donc de plus en plus nécessaire.

De plus, Shor a découvert en 1994 un algorithme quantique permettant de résoudre les problèmes de factorisation et du logarithme discret en temps polynomial. Les progrès techniques commencent aujourd'hui à s'accumuler et l'hypothèse de la « faisabailité » d'un ordinateur quantique dans les prochaines décennies semble de plus en plus probable. Ces nouveaux éléments sont donc un nouvel argument pour le développement d'une nouvelle cryptographie dite post-quantique, c'est à dire plus sûre que RSA, le problème du logarithme discret et résistante à l'ordinateur quantique. L'ajout de *post* est là pour souligner que cette dernière doit être pensée et implémentée en terme classique, par opposition à la cryptographie quantique qui ne pourra être utilisée qu'avec des ordinateurs quantiques.

Il y a aujourd'hui deux pistes majeures pour une cryptographie post-quantique : les réseaux et les codes correcteurs. Nous nous intéresserons ici qu'aux seconds, les codes correcteurs. La cryptographie se fondant sur ces derniers a été pensé par McEliece en 1978, presqu'au même moment que RSA. Cette dernière est alors tombée en désuétude car la taille des clefs est trop grosse (en revanche le chiffrement et déchiffrement sont plus rapides qu'avec RSA). L'idée fondatrice de McEliece fut d'utiliser comme clef secrète un code correcteur très structuré, pour lequel on connaît un algorithme de décodage rapide, et de fournir comme clef publique un code équivalent, construit de façon à dissimuler la structure algébrique pour qu'il se comporte comme un code aléatoire. Il y a alors deux types de cryptanalyse : chercher à retrouver la structure du code ou essayer de trouver des algorithmes de décodages *générique* efficaces. Ces deux types d'attaques mènent à des algorithmes exponentiels en la longueur du code. Aujourd'hui la croyance générale veut qu'il n'existe pas d'algorithmes polynomiaux, le problème sous-jacent étant NP-complet. Cependant il est important d'étudier les algorithmes attaquant les crypto-systèmes à base de codes correcteurs. En effet, c'est leur étude qui permettra d'ajuster au mieux les paramètres de sécurité.

Le problème étudié, la contribution proposée et les arguments en faveur de sa validité

L'objet de mon stage, sous la direction de M.Tillich, fut la cryptanalyse des systèmes à base de codes correcteurs dans le modèle « non structuré » : le code du crypto-système attaqué est uniquement supposé aléatoire, la seule structure dont nous disposons est vectorielle. Nous avons en particulier étudié un algorithme jusque-là méprisé par la communauté scientifique : le décodage statistique. Cet algorithme n'appartient pas à la famille des algorithmes les plus étudiés, dits par ensemble d'information. Une étude rigoureuse nous a permis de montrer que ce dernier est bien meilleur que les algorithmes par ensemble d'information dans « certaines zones » non négligeables. Pour cela nous avons utilisé des outils mathématiques statistiques ainsi que de l'algorithmique jusque-là très étudié. L'objectif de ce rapport est de vous présenter l'algorithme de décodage statistique et de démontrer rigoureusement son efficacité jusque-là inconnue.

Le bilan et les perspectives

Les résultats que nous avons obtenu lors de notre étude *approfondie* du décodage statistique sont dignes d'intérêt. Nous sommes donc en pleine rédaction d'un article pour la revue IEE TRANSATCTIONS ON INFORMATION THEORY où nous avons bon espoir de publier. En effet, le décodage statistique est meilleur pour certaines zones de rendement que les algorithmes de décodage générique les plus étudiées depuis près de 40 ans ! Dans le cadre de l'appel du NIST en 2016, dont l'un des buts est de normaliser les paramètres de sécurité de la cryptographie post-quantique, ce résultat sera à prendre en compte.

De plus, nous sommes aujourd'hui convaincus que l'algorithme de décodage statistique peut encore être amélioré. Nous n'avons pas encore exploité certaines idées qui semblent prometteuses. La priorité actuellement est la rédaction de notre article pour IEEE. Le début de ma thèse, sous la direction de Jean-Pierre Tillich, sera consacré à la poursuite de l'étude du décodage statistique. Le laboratoire envisage même de recruter un stagiaire courant 2017 afin d'implémenter cet algorithme.

Remerciements

Je tiens à remercier mon encadrant (et futur maître de thèse), Jean-Pierre Tillich pour sa pédagogie, sa patience, les nombreuses connaissances et intuitions qu'il a pu me donner ainsi que pour sa légendaire bonne humeur.

Je suis tout aussi reconnaissant envers Nicolas Sendrier qui m'a donné beaucoup de son temps (et avec plaisir) pour m'expliquer en détails de nombreux algorithmes.

Anne Canteaut qui malgré des débuts difficiles m'a offert sa confiance et qui m'a permis d'obtenir un financement de thèse. Je n'oublie pas non plus tous les membres de l'équipe SECRET pour l'ambiance générale qu'ils contribuent à entretenir et qui rend plaisant le travail au projet.

Je tiens aussi à remercier Ghazal Kachigar et Julia Chaulet. Ghazal qui a pris de son temps pour m'expliquer le langage Sage et Julia avec qui j'ai partagé un bureau durant six mois, sans sa bonne humeur je n'aurais pas pris tant de plaisir à étudier et faire mes recherches (et des pauses cigarettes).

1 Système de chiffrement à base de codes correcteurs

Je décris ici le système de chiffrement à clef publique présenté par McEliece en 1978. Je commencerai par décrire son fonctionnement et reviendrai ensuite sur les hypothèses que nous ferons pour le cryptanalyser ainsi que les deux modèles d'attaque. Nous renvoyons à l'annexe B pour des rappels sur les codes correcteurs et codes correcteurs aléatoires.

1.1 Le crypto-système de McEliece

Le système de McEliece utilise comme clef secrète un code linéaire binaire $\mathcal{C}$, de longueur n et de dimension k *pour lequel on connaît un algorithme rapide de décodage*. On notera t un entier inférieur à la capacité de correction du code.

Le fonctionnement de ce système de chiffrement est résumé à la table 1.

Clef secrète :
- $\mathcal{C}$ un code linéaire binaire $[n, k, d]$ choisi parmi une famille de code Γ représenté par une matrice génératrice $G \in \mathbb{F}_2^{k\times n}$
- Un algorithme de décodage pour $\mathcal{C}$
- $S \in \mathcal{G}L_k(\mathbb{F}_2)$
- $P \in \mathcal{G}L_n(\mathbb{F}_2)$ matrice de permutation

Clef publique :
- $G' = SGP$
- t entier inférieur à la capacité de correction du code

Algorithme de chiffrement :
Texte chiffré c correspondant à un message clair m : $c = mG' + e$
où $e \in \mathbb{F}_2^n$ de poids t

Algorithme de déchiffrement :
On calcule $cP^{-1} = mSG + eP^{-1}$ puis on utilise l'algorithme de décodage pour retrouver mS que l'on multiplie ensuite par S^{-1}.

TABLE 1 – Système de chiffrement à clef publique de McEliece

En réalité, la clef publique du système n'est autre qu'une matrice génératrice d'un code linéaire $\mathcal{C}'$ équivalent au code $\mathcal{C}$. On appelle $\mathcal{C}'$ *code public* par opposition au *code secret*, $\mathcal{C}$. McEliece impose cette opération dans son système pour « cacher » la structure du code secret. Intuitivement le code public va se comporter comme un code aléatoire. Un message chiffré par ce système c, s'écrit donc sous la forme d'un mot du code public dont t positions sont erronées. Retrouver le texte clair, ou de façon équivalente le vecteur d'erreur e, revient donc à résoudre le problème qui suit :

Problème 1 (Problème de décodage à distance w)**.**
Entrée : $G \in \mathbb{F}_2^{k\times n}$, $x \in \mathbb{F}_2^n$
Problème : Trouver s'il existe $m \in \mathbb{F}_2^k$ *tel que* $d_H(x, mG) \leq w$.

Si dans l'entrée nous considérons la matrice G comme quelconque, ce problème a été prouvé NP-complet en 1978 dans [3]. La difficulté à cryptanalyser le système de McEliece repose sur la difficulté de ce problème.

1.2 Hypothèses de cryptanalyse

Il y a deux possibilités naturelles pour cryptanalyser le système de McEliece. La première consiste à retrouver, à partir de la matrice publique G', la structure du code secret $\mathcal{C}$ et donc l'algorithme de décodage.

Ce type d'attaque est dite *structurelle*. Dans la pratique, la famille de codes utilisée est celle des codes de Goppa, ces derniers jouissant de bonnes propriétés statistiques. L'opération de permutation des lignes et des colonnes donne des matrices G' indiscernables de matrices aléatoires. L'autre option d'attaque consiste à chercher à décoder en utilisant uniquement la structure vectorielle de $\mathcal{C}'$ qui est vu comme un code aléatoire. On appelle ce type d'attaque *décodage générique*. Autrement dit, on cherche à résoudre le problème de la définition 1. Ce problème étant NP-complet, il y a peu d'espoir de trouver des algorithmes polynomiaux. Aujourd'hui les algorithmes de décodage les plus étudiés découlent tous du même algorithme inventé par Prange en 1968. On range ces algorithmes dans la catégorie dite *d'ensemble d'information* (ISD).

Mon stage a porté, comme souligné dans l'introduction, sur l'attaque du crypto-système de McEliece afin d'ajuster au mieux les paramètres de sécurité. Je me place dans le cadre du décodage générique. Je ferai donc dans la suite l'hypothèse :

La matrice publique est aléatoire

Les algorithmes étant exponentiels, nous nous intéresserons à l'exposant de ces derniers en négligeant les facteurs polynomiaux. Les études de complexité se feront donc toujours à facteur polynomial près. Pour cette raison nous utiliserons la notation suivante pour $f, g : \mathbb{N} \to \mathbb{R}$:

$$f = \tilde{O}(g) \iff \exists P \in \mathbb{R}[X] \ : \ f = O(P \cdot g)$$

L'étude des ces algorithmes a pour objectif de fixer des paramètres de sécurité. Nous en ferons donc une étude asymptotique en le paramètre de sécurité, c'est à dire $n \to +\infty$. Cette hypothèse est raisonnable car la puissance des ordinateurs grandissant, nous sommes actuellement devant des tailles de n suffisamment grandes pour faire nos développements asymptotiques.

2 Décodage statistique

2.1 Introduction

Cet algorithme de décodage générique a été présenté pour la première fois par A Kh. Al Jabri dans [8]. Son idée peut être décrite de la façon suivante : on se donne un code aléatoire binaire $\mathcal{C}$ de paramètre $[n, Rn]$ (R rendement du code) avec une matrice génératrice G, un mot bruité $y = xG + e$, une équation de parité h et on calcule $\langle y, h\rangle$ qui est égal à $\langle e, h\rangle$ (le produit scalaire sera toujours considéré modulo 2). Autrement dit, avec une équation de parité h il semble que l'on puisse obtenir de l'information sur le vecteur erreur e. Imaginons que nous disposions d'une telle équation h, de poids 1 avec $h_{i_0} = 1$, alors selon que $\langle y, h\rangle$ est égale à 1 ou 0, on peut tout de suite conclure quant à la présence d'une erreur en position i_0. Malheureusement la théorie des codes aléatoires nous apprend que la probabilité de trouver une telle équation est négligeable. Essayons maintenant d'élargir notre raisonnement en considérant aussi le vecteur d'erreur e. La donnée de son poids est publique dans le crypto-système de McEliece. On se donne une position quelconque et on cherche à savoir si e_i est égale à 1 ou 0. Donnons-nous une équation de parité h telle que $h_i = 1$. Alors pour que $\langle e, h\rangle$ soit égal à 1 il faut qu'un nombre pair ou impair de positions du support de h, selon que e_i est égal à 1 ou 0, coïncident avec des positions du support de e. On remarque alors que ce nombre de positions pair ou impair s'inverse pour que $\langle y, h\rangle$ soit égal à 0. On en déduit que la probabilité sur e d'avoir $\langle y, h\rangle = 1$ si e_i est égale à 1 est la même que d'obtenir $\langle y, h\rangle = 0$ si e_i est égale à 0. On peut donc s'attendre à observer un biais dans la distribution des $\langle y, h\rangle$

L'objectif des deux premières sous-sections est de formaliser cette intuition et de la démontrer. Dans la troisième sous-section nous présenterons formellement l'algorithme de décodage statistique et nous donnerons sa complexité asymptotique.

Dans la suite on désignera par $\mathcal{C}$ un code aléatoire binaire de paramètre $[n, Rn]$. De plus on utilisera les notations suivantes :

· $\mathcal{H}_w$ désigne un sous-ensemble de $\{h \in \mathcal{C}^\perp : \ w_H(h) = w\}$ où w_H désigne le poids de Hamming ;
· $\mathcal{H}_{w,i}$ désigne un sous-ensemble de $\mathcal{H}_w \cap \{m \in \mathbb{F}_2^n \ : \ m_i = 1\}$.

Remarque 1. *Nous ferons dans cette section de nombreux calculs probabilistes sur les codes dits « aléatoires ». Nous renvoyons le lecteur à l'annexe B pour des justifications plus précises.*

2.2 Modèle combinatoire

Dans le crypto-système de McEliece, le poids de l'erreur est fixé à t. Nous allons donc désigner par $\mathcal{T}$ l'ensemble des lois uniformes $\mathcal{U}(x \in \mathbb{F}_2^n : w_H(x) = t)$. L'écriture $\mathbb{P}_{e\sim\mathcal{T}}$ signifiera que l'on tire les vecteurs e de façon uniforme dans l'univers des mots de poids t. D'après la note introductive, on fixe un vecteur $h \in \mathcal{H}_{w,i}$ et on s'intéresse aux probabilités à priori :

$$q^+_{w,h,i} = \mathbb{P}_{e\sim\mathcal{T}}\left(\langle h, e\rangle = 1 | e_i = 1\right)$$

$$q^-_{w,h,i} = \mathbb{P}_{e\sim\mathcal{T}}\left(\langle h, e\rangle = 1 | e_i = 0\right)$$

Un calcul simple nous donne :

$$q^+_{w,h,i} = \frac{\sum\limits_{j=0,\ pair}^{t-1} \binom{w-1}{j} \cdot \binom{n-w}{t-1-j}}{\binom{n}{t}}$$

$$q^+_{w,h,i} = \frac{\sum\limits_{j=0,\ impair}^{t-1} \binom{w-1}{j} \cdot \binom{n-w}{t-j}}{\binom{n}{t}}$$

Ces calculs ne nous permettent pas de conclure quant à l'existence d'un biais dans la distribution des $\langle y, h\rangle$, contrairement à ce que nous affirmions dans l'introduction, pour h parcourant $\mathcal{H}_{w,i}$. Cependant comme nous l'avons souligné dans la section qui précède, l'étude des algorithmes de décodage prend tout sons sens dans le cas asymptotique. Or une bonne façon d'approximer le modèle combinatoire dans ce cas est d'utiliser le modèle probabiliste du canal BSC(p) (nous renvoyons à l'annexe B pour une définition précise de ce modèle). En effet, si nous choisissons $p = \frac{t}{n}$, alors toutes les erreurs vont être asymptotiquement en n de poids $t(1 + o(1))$. De cette façon nous approchons le modèle combinatoire et comme nous le verrons les calculs se simplifient.

2.3 Modèle du canal binaire symétrique

On considère le modèle BSC(p) pour $p \in [0, \frac{1}{2}[$. Nous rappelons que $e \sim \text{BSC}(p)$ signifie que $e = (e_1, \cdots, e_n)$ où les e_i sont des variables aléatoires de Bernoulli indépendantes et de paramètre p.

On fixe $h \in \mathcal{H}_{w,i}$ et soient $\alpha^+_{w,i,h}$ et $\alpha^-_{w,i,h}$ les deux probabilités à priori :

$$\alpha^+_{w,i,h} = \mathbb{P}_{e\sim BSC(p)}\left(\langle e, h\rangle = 1 | e_i = 1\right)$$

$$\alpha^-_{w,i,h} = \mathbb{P}_{e\sim BSC(p)}\left(\langle e, h\rangle = 1 | e_i = 0\right)$$

Un calcul simple donne :

$$\alpha^+_{w,i,h} = \frac{1}{2} + \frac{(1-2p)^w}{2} \quad ; \quad \alpha^-_{w,i,h} = \frac{1}{2} - \frac{(1-2p)^w}{2}$$

Remarque 2. *Ces deux probabilités ne dépendent ni de $h \in \mathcal{H}_{w,i}$ ni de i. Dans la suite on les notera donc α^+_w et α^-_w.*

On pose

$$\varepsilon_w = \frac{(1-2p)^w}{2}$$

Remarque 3. *Dans notre étude asymptotique du décodage statistique nous aurons $\varepsilon_w \to 0$ avec $n \to +\infty$.*

Comme $\varepsilon_w > 0$, on a l'inégalité $\alpha^+_w > \alpha^-_w$. Autrement dit si $e_i = 1$, le produit scalaire $\langle y, h\rangle$ aura tendance à donner plus souvent 1 que dans le cas $e_i = 0$. Il y a donc bien un biais dans la distribution

des $\langle y, h\rangle$. Notre objectif est donc de construire un « distingueur ». Cet outil sera à la base du décodage statistique. On commence par définir la variable aléatoire pour j éléments $\{h^1, \cdots, h^j\}$ de $\mathcal{H}_{w,i}$:

$$V_i^j = \sum_{k=1}^{j} \langle y, h^k\rangle = \sum_{k=1}^{j} \langle e, h^k\rangle \in \mathbb{Z}$$

Pour discerner si e_i est égale à 1 ou 0, nous allons utiliser un test statistique : le test de vraissemblance. Pour cela nous commençons par définir les deux hypothèses :

$$\mathcal{H}_0 \ : \ e_i = 0 \quad ; \quad \mathcal{H}_1 \ : \ e_i = 1$$

Notre objectif est de déterminer la bonne hypothèse. De façon usuelle on distingue les deux erreurs possibles dites de non-détection et fausse alerte. La situation se résume par le tableau :

vraie situation	conclusion	
	accepte $\mathcal{H}_0$	accepte $\mathcal{H}_1$
$e_i = 0$	pas d'erreur	type-I erreur (fausse alerte)
$e_i = 1$	type-II erreur (non-détection)	pas d'erreur

Les variables aléatoires $\langle e, h^k\rangle$ sont *indépendantes* et uniformément distribuées. Sous l'hypothèse $\mathcal{H}_0$, $\langle e, h^k\rangle \sim \mathcal{B}(1/2 - \varepsilon_w)$ et sous l'hypothèse $\mathcal{H}_1$, $\langle e, h^k\rangle \sim \mathcal{B}(1/2 + \varepsilon_w)$.

Nous allons utiliser la proposition qui suit pour construire notre « distingueur »

Proposition 1 (Borne de Chernoff)**.** *Soit* $0 < p < 1$, $Y_1, \cdots, Y_j$ *i.i.d* $\sim \mathcal{B}(p)$ *et* $S_j = \sum_{k=1}^{j} Y_k$. *Alors*

$$\forall \varepsilon \geq 0, \quad \mathbb{P}\left(|S_j - jp| \geq j\varepsilon\right) \leq 2 \cdot 2^{-2j\varepsilon^2/\ln(2)}$$

Dans ce qui suit on négligera le facteur constant 2. On en déduit alors que :

— Sous $\mathcal{H}_0$, nous avons :

$$\mathbb{P}_{e\sim BSC(p)}\left(|V_i^j - j(1/2 - \varepsilon_w)| \geq j\varepsilon_w\right) \leq 2^{-2j\varepsilon_w^2/\ln(2)}$$

— Sous $\mathcal{H}_1$, nous avons :

$$\mathbb{P}_{e\sim BSC(p)}\left(|V_i^j - j(1/2 + \varepsilon_w)| \geq u\varepsilon_w\right) \leq 2^{-2j\varepsilon_w^2/\ln(2)}$$

Pour prendre notre décision nous procédons de la façon suivante : si $V_i^j < \frac{j}{2}$, nous choisissons $\mathcal{H}_0$ et $\mathcal{H}_1$ sinon. De cette façon si nous prenons un échantillon de taille $j \sim \frac{-\ln(\varepsilon_w)}{\varepsilon_w^2}$, notre procédé mène à une erreur (type-I ou type-II) avec une probabilité tendant asymptotiquement vers 0 (on a $\varepsilon_w \to 0$ quand $n \to +\infty$). Nous pouvons résumer la situation avec la figure 1.

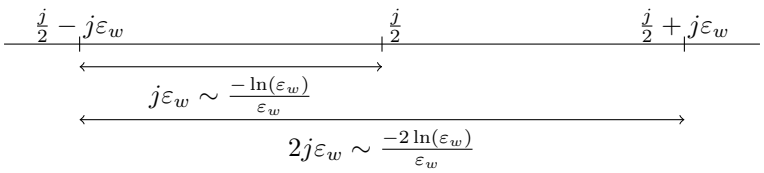


FIGURE 1 – Région de décision. Le côté gauche correspond à l'hypothèse $\mathcal{H}_0$ tandis que la région de droite mène à l'hypothèse $\mathcal{H}_1$

Nous avons donc construit un distingueur asymptotique pour discerner deux variables de Bernoulli de paramètre distant de deux ε_w avec $\varepsilon_w \to 0$. Pour cela nous avons besoin d'un échantillon d'une taille de l'ordre de $\frac{-\ln(\varepsilon_w)}{\varepsilon_w^2}$. On peut alors se demander, de façon légitime, existe t-il un meilleure procédé ? Autrement dit, peut-on diminuer la taille de l'échantillon pour prendre notre décision ? La réponse est donnée par la théorie des types discrets, et c'est celle-ci est négative. Le théorème de Sanov dont nous rappelons l'énoncé et la preuve dans l'annexe D nous apprend que l'exposant de la borne de Chernoff est asymptotiquement optimal. Notre test statistique ne peut donc être amélioré. De plus lemme de Neymann-Pearson (dont on trouvera aussi une preuve dans l'annexe D) nous permet d'affirmer que nous avons le meilleur test. La région de décision par vraissemblance est optimale dans le sens où toute autre région de décision mènera à des probabilités d'erreur plus grandes.

Maintenant que nous disposons de notre distingueur optimal nous passons à la présentation du décodage statistique.

2.4 L'algorithme de décodage statistique

Nous reprenons dans cette sous-section les notations et résultats qui précèdent. Comme nous l'avons déjà souligné, l'algorithme de décodage statistique va utiliser le biais dans la distribution des $\langle y, h\rangle$. Dans le crypto-système de McEliece, le nombre d'erreur t ajouté au message est connu. Afin de pouvoir utiliser les résultats de la sous-section 2.3 et d'après les considérations de la sous-section 2.2, nous fixons le paramètre

$$p = \frac{t}{n}$$

Asymptotiquement le poids des erreurs sera donc bien t. De plus pour un poids $w \in \mathbb{N}$ fixé on a :

$$\frac{-\ln(\varepsilon_w)}{\varepsilon_w^2} = \tilde{O}\left(2^{-2w\cdot\log_2(1-2\frac{t}{n})}\right)$$

Dans la suite P_w désignera le nombre d'équations $2^{-2w\cdot\log_2(1-2\frac{t}{n})}$. On fixe un indice i. L'objectif d'un algorithme de décodage générique est de déterminer si e_i est égale à 1 ou 0. Or d'après ce qui précède on peut distinguer ces deux cas si on dispose de P_w équations de parité dans $\mathcal{H}_{w,i}$. L'idée informelle du décodage statistique est donc dans un premier temps de trouver P_w équations de parité h^k de $\mathcal{H}_{w,i}$ et de calculer ensuite :

$$V_i = \sum_{k=1}^{P_w} \langle y, h^k\rangle$$

et selon que V_i est supérieur ou inférieur à $P_w/2$ en déduire que e_i égale à 1 ou 0.

Nous allons maintenant présenter, de façon plus formelle, l'algorithme de décodage statistique. On note $\texttt{CalculParité}_w(G, i)$ l'algorithme qui prend comme entrée la matrice génératrice du code $\mathcal{C}$ et renvoie P_w équations de $\mathcal{H}_{w,i}$.

`DecoStat` : Décodage Statistique

Input : $G \in \mathbb{F}_2^{k\times n}$, $y = xG + e \in \mathbb{F}_2^n$, $w \in \mathbb{N}$
Output : e /**Vecteur d'erreur**/
for $i = 1 \cdots n$ **do**
    $\mathcal{S}_i \leftarrow \texttt{CalculParité}_{P_w}(G, i)$ /**Algorithme auxiliaire**/
    $v_i \leftarrow 0$ /**compteur initialisé à 0**/
    for all $h_i \in \mathcal{S}_i$ **do**
        $v_i \leftarrow v_i + \langle y, h_i\rangle$
    end for
    if $v_i < \frac{P_w}{2}$ **then**
        $e_i \leftarrow 0$
    else
        $e_i \leftarrow 1$
    end if

`DecoStat` est un algorithme probabiliste. Il fournira asymptotiquement le bon résultat avec une probabilité proche de 1 d'après ce qui précède (on a assez d'équations de parité). De plus sa complexité est (à un facteur polynomial près) :

$$O\left(P_w\right) + O\left(|\texttt{CalculParité}_{P_w}|\right)$$

Nous allons maintenant nous intéresser à l'algorithme $\texttt{CalculParité}_{P_w}$. Pour cela nous commençons par décrire un algorithme probabiliste calculant suffisamment d'équations pour $w = \frac{Rn}{2}(1 + o(1))$ (le petit o étant sur n).

Nous renvoyons à l'annexe B.1.2 pour des détails sur le pivot de Gauss

---

$\texttt{CalculParité}_{P_{Rn/2(1+o(1))}}$: **Calcul d'équations de parité de poids** $\frac{Rn}{2}(1 + o(1))$

1: Input : $G \in \mathbb{F}_2^{k\times n}$, $i \in \mathbb{N}$
2: Output : $\mathcal{S}$ /*$P_{Rn/2(1+o(1))}$ *équations de parité**/
3: $\mathcal{S} \leftarrow [\,]$
4: On choisit un ensemble de taille Rn et on met à l'aide d'un pivot de Gauss G sous la forme :

$$[I_{Rn} \mid G_1]$$

5: On calcule une matrice de parité :

$$H = [I_{n(1-R)} \mid {}^tG_1]$$

6: On ajoute à $\mathcal{S}$ les lignes de H de poids $\leq \frac{Rn}{2}$ et de coordonnée i non nulle
7: **while** $|\mathcal{S}| < P_{Rn/2}$ **do**
8: On reprend à la ligne 3
9: **end while**=0

On a supposé que G, donc H est une matrice aléatoire (cf 1.2). Les lignes de H auront donc un poids typique de $\frac{Rn}{2}(1+o(1))$ (on renvoie à l'annexe F pour cette affirmation qui repose sur la borne de Bienaymé-Tchebychev). On en déduit qu'asymptotiquement l'algorithme nous renvoie des équations de bon poids. Un pivot de Gauss ayant une complexité polynomiale, ce que l'on néglige ici, on en déduit que la complexité de $\texttt{CalculParité}_{P_{Rn/2(1+o(1)}}$ est $\tilde{O}(P_{Rn/2(1+o(1))})$. De plus cet algorithme fournit bien $P_{Rn/2(1+o(1))}$ équations de parité. Donc la complexité du décodage statistique est pour cet algorithme auxiliaire :

$$\tilde{O}\left(\frac{-\ln(\varepsilon_{Rn/2(1+o(1))})}{\varepsilon^2_{Rn/2(1+o(1))}}\right) = \tilde{O}\left(2^{-Rn\log_2(1-2\frac{t}{n})}\right)$$

Nous avons donc une complexité exacte pour le décodage statistique et nous pouvons déjà en déduire un premier résultat intéressant. Nous renvoyons le lecteur à l'annexe C pour plus de détails sur les algorithmes de décodage ISD. D'après les travaux de R.Canto-Torres et N.Sendrier dans [4], tous les algorithmes de décodage par ensemble d'information ont une complexité asymptotique de l'ordre de

$$2^{-\log_2(1-R)\cdot t}$$

pour un régime d'erreur sous-linéaire, c'est à dire $t = o(n)$. L'algorithme de décodage statistique a lui sous ce régime une complexité de :

$$2^{2(R/\ln(2))\cdot t}$$

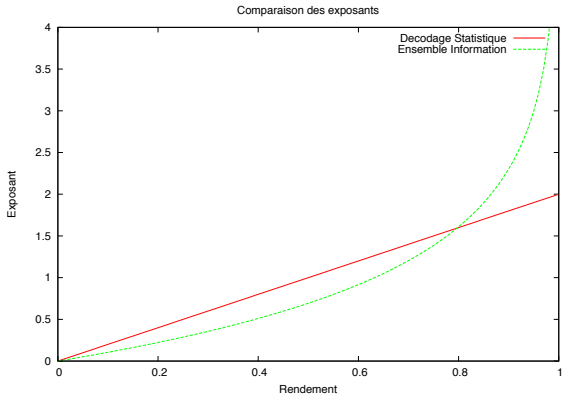
On en déduit la proposition suivante :

Proposition 2. *Dans le cas d'un régime d'erreur sous linéaire, l'algorithme de décodage statistique, utilisé avec* $\texttt{CalculParité}_{P_{Rn/2(1+o(1))}}$*, est meilleur que les algorithmes de type ISD pour des rendements supérieurs à* 0.8*.*

La figure 2 compare les exposants de ces deux types d'algorithme dans le cas sous-linéaire. Nous remarquons alors que le décodage statistique offre une amélioration asymptotique importante à partir d'un

rendement égal à 0.8. Ce premier résultat a son importance car le régime sous-linéaire est « réaliste ». Par exemple pour les codes de Goppa le paramètre d'erreur t est de l'ordre de $\frac{n}{\log_2(n)}$.

FIGURE 2 – Comparaison des exposants asymptotiques du décodage statistique et ISD



Une fois ce premier résultat obtenu, nous avons cherché à trouver des algorithmes de calcul d'équations de parité pour $w < \frac{Rn}{2}$ dans le cas sous-linéaire. En effet, plus le poids des équations de parité est faible, moins il est nécessaire d'en avoir pour prendre notre décision ($w \mapsto P_w$ est croissante). La complexité du décodage statistique, sans la phase de calcul des équations, étant donné par le nombre d'équations nécessaires, sa complexité diminuera avec $w < \frac{Rn}{2}$. Pour cela nous avons pensé à des techniques dites de type *paradoxe des anniversaires* (chercher des collisions à travers les coordonnées pour faire diminuer le poids). Il s'avère que toutes ces améliorations ont un coût exponentiel pour diminuer le poids. Or dans le régime sous-linéaire, le décodage statistique a une complexité sous-exponentielle. Nous n'avons donc pas abouti avec ces idées.

Les diverses variantes de l'algorithme de Prange sont asymptotiquement meilleures dans un régime plus classique, celui où le poids de l'erreur est donné par la borne de Gilbert-Varshamov : $\tau_{GV} = h^{-1}(1-R)$ (h fonction entropie). Il s'avère que cette borne donne le nombre d'erreurs le plus difficile à corriger tout en restant possible pour les codes aléatoires (cf annexe C). Faire l'hypothèse d'un nombre d'erreur sur cette borne est donc cohérent et réaliste. Dans la suite nous remplacerons donc $\frac{t}{n}$ par τ_{GV} et nous verrons que les améliorations apportées à l'algorithme de Prange nous permettent d'améliorer de façon significative l'algorithme de décodage statistique.

3 Amélioration du décodage statistique

Avant de présenter les diverses améliorations que nous avons faites sur le décodage statistique, nous revenons sur le problème du calcul d'équations de parité afin de bien fixer le cadre d'étude.

Pour un code $\mathcal{C}$ engendré par une matrice G de paramètre $[n, Rn]$ nous avons vu comment calculer des équations de parité de poids approximativement $\frac{Rn}{2}$ en temps amorti $\tilde{O}(1)$. Pour ce faire nous avons procédé de la façon suivante :

— On calcule une matrice de parité H de $\mathcal{C}$ que l'on met (à permutation près) sous une forme systématique :

$$\begin{bmatrix} I_{n(1-R)} & \tilde{H} \end{bmatrix} \text{ où } \tilde{H} \in \mathbb{F}_2^{n(1-R)\times Rn}$$

— On tire aléatoirement des lignes de cette matrice et on ne garde que celles de poids $\leq \frac{Rn}{2}$. Ces dernières comme nous l'avons vu auront un poids de $\frac{Rn}{2}(1+o(1))$.

Cet algorithme a une complexité polynomiale en n et fournit une équation de parité de poids $\frac{Rn}{2}(1+o(1))$. Étant donné que dans notre modèle asymptotique nous négligeons les facteurs polynomiaux, nous pouvons affirmer que les équations de parité ont été obtenu en temps amorti 1. Nous insistons sur ce point de temps amorti car la complexité du décodage statistique (sans la phase de calcul des équations de parité) est le nombre d'équations nécessaires à la décision. Nous pouvons résumer la situation de la façon suivante :

— $\mathcal{D}$ désigne l'algorithme de décodage statistique sans la phase de calcul des équations ;
— $\mathcal{A}$ désigne un algorithme probabiliste renvoyant pour une entrée $\mathcal{C}$, N équations en temps T. Si on relance $\mathcal{A}$ sur la même entrée il peut renvoyer d'autres équations du code.

Supposons que le décodage statistique a besoin de P équations pour un code $\mathcal{C}$. La complexité totale sera donnée à un facteur polynomial près par :

$$O(P) + O\left(T \cdot \max(1, \frac{P}{N})\right)$$

$\rightarrow$ Si $\mathcal{A}$ fournit des équations en temps amorti $\tilde{O}(1)$, cela signifie que $\frac{T}{N} = 1$.

Donc si on suppose que $N \leq P$, la complexité est alors donnée par le nombre d'équations

Comme nous l'avons souligné dans la section qui précède, la complexité du décodage statistique est directement liée au poids des équations : plus les équations ont un poids faible, plus la complexité sera faible. L'objectif de cette section est de trouver des algorithmes permettant de calculer des équations de poids strictement inférieur à $\frac{Rn}{2}(1+o(1))$. Le schéma qui précède ne tient pas compte du poids, nous allons l'adapter et expliquer de façon informelle ce qui se produit.

— $\mathcal{D}_w$ désigne l'algorithme de décodage statistique (sans la phase de pré-calcul) avec des équations de parité de poids w. Sa complexité sur un input $\mathcal{C}$ est donnée par un nombre d'équations de parité P_w ;
— $\mathcal{A}_w$ désigne un algorithme probabiliste renvoyant N_w équations de poids w en temps T_w.

Si $N_w \leq P_w$, la complexité du décodage statistique est donnée à un facteur polynomial près par :

$$O\left(T_w \cdot \frac{P_w}{N_w}\right) + O\left(P_w\right)$$

D'après la section qui précède $w \mapsto P_w$ est une fonction croissante alors que les algorithmes que nous allons considérer sont tels que $w \mapsto T_w$ est une fonction décroissante. La discussion qui précède nous a invité à considérer des algorithmes $\mathcal{A}_w$ fournissant des équations en temps amorti $\tilde{O}(1)$. Nous nous fixerons cette contrainte mais avant cela il ne faut pas oublier que N_w doit être inférieur à P_w. On résume alors la situation par les deux contraintes :

· $1^{ère}$ **contrainte :** $N_w \leq P_w$

· $2^{ème}$ **contrainte :** temps amorti de $\mathcal{A}_w$ de $\tilde{O}(1)$.

Sous ces contraintes nous allons donc chercher à minimiser P_w, autrement dit à trouver des algorithmes $\mathcal{A}_w$ vérifiant les deux contraintes pour w de plus en plus faible. Il faut bien comprendre ici que plus w sera faible, plus un algorithme $\mathcal{A}_w$ sera difficile à trouver sous les contraintes. Il existe des algorithmes pour trouver des équations de poids très faibles, en revanche ces derniers ne vérifient pas la deuxième contrainte car une équation aura un coût exponentiel.

Rappel : $P_w = 2^{-2 \cdot w \cdot \log_2(1-2\cdot\tau_{GV})}$.

Les trois sous-sections qui suivent présentent trois algorithmes de type $\mathcal{A}_w$ vérifiant nos contraintes. Nous les présentons par ordre de difficulté croissante. Le premier algorithme que nous présentons est celui

de Dumer. Ce dernier utilise une technique dite par *paradoxe des anniversaires*. Les deux algorithmes qui suivront chercheront tous à optimiser le *paradoxe des anniversaires*, chacun en introduisant un degré de liberté supplémentaire. Il est important de bien comprendre les calculs de complexité de l'algorithme de Dumer car dans les sous-sections qui suivront, nous les utiliserons implicitement.

Remarque 4. *Dans nos calculs de complexité nous négligeons les facteurs polynomiaux (cf 1.2)*

3.1 L'amélioration de Dumer

L'algorithme que nous venons de décrire pour trouver des équations de parité ne consiste qu'en un pivot de Gauss puis une recherche probabiliste sur les lignes. Cette démarche est exactement celle qu'eut Prange pour résoudre le problème du décodage (voir Annexe C). Au fil des années, de nombreuses améliorations ont été faites dans le but de réduire la complexité de cet algorithme, donnant alors naissance à la famille des algorithmes dits par *ensemble d'information*. Nous nous sommes intéressés à ces diverses améliorations en essayant de les adapter à notre problème. La première d'entres elles est dûe à Dumer. L'idée générale est de résoudre un sous-problème à l'aide du *paradoxe des anniversaires*. Nous allons ici décrire cet algorithme, calculer sa complexité et montrer l'amélioration numérique qu'il apporte au décodage statistique.

Nous allons commencer par donner le principe général de notre algorithme. Les améliorations que nous présenterons concerneront toujours le point 1.

On commence par mettre la matrice G (à permutation près) sous une forme systématique partielle :

$$\begin{bmatrix} I_{Rn-l} & G_1 \\ 0 & G_2 \end{bmatrix} \text{ où } G_1 \in \mathbb{F}_2^{(Rn-l)\times(n(1-R)+l)} \text{ et } G_2 \in \mathbb{F}_2^{l\times(n(1-R)+l)}$$

1. On résout le problème $\mathrm{CSD}(G_2, p, 0_{[l]})$.
2. Pour chaque solution e obtenu en 1. on renvoie le vecteur $e_s = (G_1 e^t, e)$.

Remarque 5. *On rappelle que résoudre CSD($G_2, p, , 0_{[l]}$) revient à trouver une combinaison de p colonnes de G_2 donnant le vecteur nul.*

· **Correction :** On rappelle qu'une équation de parité du code $\mathcal{C}$ est un vecteur h tel que $Gh^t = 0$. Notre algorithme nous donne bien des solutions car $Ge_s^t = (G_1e^t + G_1e^t, G_2e^t) = (0, 0)$.

· **Nombre de solutions :** Le nombre de solutions que nous avons obtenu est donné par le nombre de vecteurs obtenus en 1. De plus la complexité de cet algorithme est la complexité de 1. car l'étape 2. est polynomial en n, proportion que l'on néglige.

Remarque 6. *Il se peut que nous n'ayons pas assez de solutions à la fin de l'algorithme. Dans ce cas on le relance avec un autre pivot de Gauss (cf annexe B pour plus de détails).*

· **Poids des solutions :** Dans notre problème la matrice G est considérée comme aléatoire, donc G_1 est aussi aléatoire. De plus, celle-ci possédant $Rn - l$ lignes nous obtenons asymptotiquement des équations de parité de poids :

$$p + \frac{Rn - l}{2}(1 + o(1))$$

Pour résoudre le premier point, Dumer propose de commencer par « découper » la matrice G_2 en deux :

$$G_2 = \left[G_2^{(1)} | G_2^{(2)}\right] \text{ où } G_2^{(i)} \in \mathbb{F}_2^{l\times(\frac{n(1-R)+l}{2})}$$

On construit alors les deux listes :

$$\mathcal{L}_1 = \left\{ \left(e_1, G_2^{(1)} e_1^t\right) \mid e_1 \in \mathbb{F}_2^{\frac{n(1-R)+l}{2}} \text{ et } w_H(e_1) = \frac{p}{2} \right\}$$

$$\mathcal{L}_2 = \left\{ \left(e_2, G_2^{(2)} e_2^t\right) \mid e_2 \in \mathbb{F}_2^{\frac{n(1-R)+l}{2}} \text{ et } w_H(e_2) = \frac{p}{2} \right\}$$

On les intersecte ensuite selon les secondes coordonnées en ne conservant que les premières coordonnées. Autrement dit on obtient :

$$\left\{(e_1, e_2) \mid w_H(e_i) = \frac{p}{2} \text{ et } G_2^{(1)} e_1^t = G_2^{(2)} e_2^t\right\}$$

Remarque 7. *On appelle ce procédé une fusion.*

Cet algorithme résout le point 1. Nous allons maintenant rentrer dans les détails algorithmiques afin d'en donner une bonne estimation.

Dumer (Fusion)

Input : $G_2^{(1)}, G_2^{(2)} \in \mathbb{F}_2^{l \times (\frac{n(1-R)+l}{2})}$, p
Output : $\mathcal{S}$ */ Ensemble de vecteurs résolvant 1.*/*
$\mathcal{S} \leftarrow [\,]$ /*liste vide*/
$\mathcal{T} \leftarrow [\,]$ /*table de hachage*/
for all $e_1 \in \mathbb{F}_2^{(n(1-R)+l)/2}$ de poids $p/2$ **do**
    $x \leftarrow G_1^{(1)} e_1^t$
    $\mathcal{T}[x] \leftarrow \mathcal{T}[x] \cup \{e_1\}$
end for
for $e_2 \in \mathbb{F}_2^{(n(1-R)+l)/2}$ de poids $p/2$ **do**
    $x \leftarrow G_1^{(2)} e_2^t$
    for all $e_1 \in \mathcal{T}[x]$ **do**
        $\mathcal{S} \leftarrow \mathcal{S} \cup \{(e_1, e_2)\}$
    end for
end for

Rappelons que nous négligeons tous les facteurs polynomiaux, cela conduisant notamment à négliger les calculs matriciels. En revanche la construction de la table de hachage aura un coût de $\binom{(n(1-R)+l)/2}{p/2}$. Il reste maintenant à compter la taille de $\mathcal{S}$ que l'on construit. La complexité de l'algorithme est donc :

$$\tilde{O}\left(\binom{(n(1-R)+l)/2}{p/2} + \#\mathcal{S}\right)$$

Afin d'estimer la taille de $\mathcal{S}$ nous utiliserons la proposition qui suit (dont on trouvera une preuve en annexe C.3.2 à la proposition 12) :

Proposition 3. *Soient L_1, L_2 deux listes de vecteurs de $\{0,1\}^l$ dont les entrées sont aléatoires et distribuées de façon uniforme. Alors l'espérance de la taille de la fusion est :*

$$\frac{|L_1| \cdot |L_2|}{2^l}$$

La matrice G_2 étant considérée comme aléatoire, les éléments de $\mathcal{L}_1$ et $\mathcal{L}_2$ vérifient les conditions de la proposition qui précède. On en déduit la proposition :

Proposition 4 (Complexité de Dumer (Fusion))**.**
La complexité de l'algorithme de Dumer(Fusion) est

$$\tilde{O}\left(\binom{(n(1-R)+l)/2}{p/2} + \frac{\binom{(n(1-R)+l)/2}{p/2}^2}{2^l}\right)$$

et il fournit un nombre d'équations :

$$\frac{\binom{(n(1-R)+l)/2}{p/2}^2}{2^l}$$

Notre objectif maintenant est de choisir p et l afin de satisfaire les deux contraintes. Afin d'avoir un temps amorti de 1 :

$$\frac{\binom{(n(1-R)+l)/2}{p/2}^2}{2^l} = \binom{(n(1-R)+l)/2}{p/2}$$

ce qui donne asymptotiquement en terme d'exposant (en base 2) :

$$l = \frac{n(1-R)+l}{2} \cdot h\left(\frac{p}{n(1-R)+l}\right)$$

On obtient donc

$$p = (n(1-R)+l) \cdot h^{-1}\left(\frac{2l}{n(1-R)+l}\right)$$

De plus on remarque que sous cette contrainte, 2^l est le nombre de solutions. La contrainte sur le décodage statistique devient asymptotiquement en terme d'exposant (en base 2) :

$$l \leq -2 \cdot \left(p + \frac{Rn-l}{2}\right) \cdot \log_2(1-2\tau_{GV})$$

Il est maintenant important de normaliser tous nos paramètres car ce sont eux qui donneront l'exposant asymptotique. On utilisera la notation :

$$\lambda = \frac{l}{n} \quad ; \quad \pi = \frac{p}{n}$$

Les contraintes deviennent :

$$\pi = ((1-R)+\lambda) \cdot h^{-1}\left(\frac{2\lambda}{1-R+\lambda}\right) \quad ; \quad \lambda \leq -2 \cdot \left(\pi + \frac{R-\lambda}{2}\right) \cdot \log_2(1-2\tau_{GV})$$

Afin d'estimer l'inégalité, nous allons faire un développement limité de π en $\lambda \approx 0$, ce qui est cohérent car dans la pratique le paramètre λ est proche de 0.

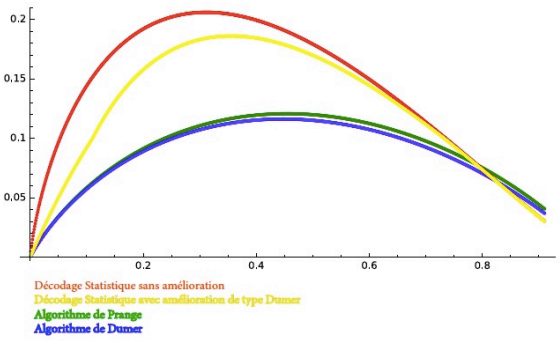
On obtient :

$$\pi = \frac{-2\lambda}{\log_2(\lambda)}(1+o(1))$$

On en déduit que $\lambda \mapsto -2 \cdot \left(\pi + \frac{R-\lambda}{2}\right) \cdot \log_2(1-2\tau_{GV})$ est décroissante au voisinage de 0. Il est donc clair que pour obtenir un exposant minimal pour le décodage statistique il faut trouver λ tel qu'il y ait égalité dans l'inégalité qui précède. J'ai donc écrit un programme en Sage (voir annexe E) qui pour chaque rendement $R \in [0,1]$ calcule un tel λ (en remplaçant π par son expression en fonction de λ) puis en déduit la poids des équations de parité que l'on a obtenu.

Dans les figures 3 et 4 nous donnons les exposants asymptotiques du décodage statistique (sans et avec l'amélioration de Dumer) que nous comparons avec ceux des algorithmes par ensemble d'information de Prange et Dumer.

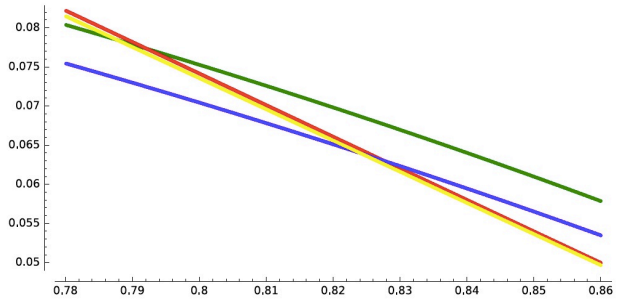
FIGURE 3 – Comparaison des exposants asymptotiques



On constate une sensible amélioration au voisinage de 0. En revanche en 0.835, là où le décodage statistique devient meilleur que Dumer, l'amélioration n'est pas flagrante. En y regardant de plus près, nous obtenons la figure 2. L'amélioration est légère mais elle n'est pas à négliger.

Remarque 8. *L'algorithme de Dumer nous a permis d'obtenir un meilleur poids pour les équations de parité. Le point de vue que nous avons adopté ne nous permet pas d'expliquer d'où vient l'idée de Dumer et en quoi elle nous permet de gagner. Nous renvoyons à l'annexe C pour une explication du gain.*

FIGURE 4 – Comparaison des exposants asymptotiques en rendement proche de 0.8



De plus, un développement limité au voisinage de $R = 0$ de l'exposant du décodage statistique avec l'amélioration de Dumer donne :

$$R + \frac{R}{\log_2(R)} + o\left(\frac{R}{\log_2(R)}\right)$$

On en déduit que la pente de l'exposant du décodage statistique est au voisinage de 0 de 1. Or celle du décodage de Dumer-ISD est de 1/2. Nous ne serons donc pas meilleur que Dumer-ISD au voisinage de 0. De plus, ce développement limité nous apprend que le décodage statistique n'est pas efficace au voisinage de 0. Asymptotiquement il a la même complexité que l'algorithme de naïf consistant à regarder tous les vecteurs de l'espace ambiant.

Nous allons maintenant passer à l'algorithme de May, Meurer et Thomae (que l'on abréviera par MMT). L'objectif est d'améliorer l'algorithme du *paradoxe des anniversaires* en introduisant un degré de liberté supplémentaire. La version que nous présentons n'est pas celle de [9] mais celle de N.Sendrier.

3.2 L'amélioration de MMT

On se donne comme précédemment la matrice génératrice du code $\mathcal{C}$ que l'on suppose être de la forme :

$$\begin{bmatrix} I_{Rn-l} & G_1 \\ 0 & G_2 \end{bmatrix} \text{ où } G_1 \in \mathbb{F}_2^{(Rn-l)\times(n(1-R)+l)} \text{ et } G_2 \in \mathbb{F}_2^{l\times(n(1-R)+l)}$$

Dans l'algorithme de Dumer on a coupé G_2 en deux puis cherché deux ensembles d'indice $I_1 \subseteq [1, (n(1-R)+l)/2[$ et $I_2 \subseteq [(n(1-R)+l)/2, n(1-R)+l]$ de taille $|I_1| = |I_2| = \frac{p}{2}$ tels que :

$$\sum_{i\in I_1} q_i + \sum_{i\in I_2} q_i = 0_{[l]} \text{ où } 0_{[l]} \text{ est le vecteur nul de } \mathbb{F}_2^l \tag{1}$$

Nous n'avons pas parlé formellement d'ensemble d'indices dans l'algorithme de Dumer. Il faut comprendre cependant que l'on peut voir les vecteurs $e_1, e_2 \in \mathbb{F}_2^{(n(1-R)+l)/2}$ que nous recherchions dans l'algorithme de fusion comme des ensembles d'indice où $Supp(e_i) = I_i$. L'équation (1) est alors équivalente à :

$$G_2(e_1, e_2)^t = 0_{[l]}$$

Dans la suite nous confondrons régulièrement ces deux points de vue.

Remarque 9. *Avec cette méthode on est sûr de ne pas avoir de « doublons » dans les solutions. En effet deux couples distincts d'ensembles* (I_1, I_2) *et* (I'_1, I'_2) *donneront toujours des solutions distinctes.*

La remarque que nous venons de faire est à l'origine de la technique dite *des représentations* que l'algorithme MMT utilise.

L'idée de l'algorithme de MMT est cette fois de chercher les collisions pour des vecteurs $e_i \in \mathbb{F}_2^{n(1-R)+l}$ de poids $\frac{p}{2} + \varepsilon$ où ε est choisi tel que :

$$\varepsilon = \frac{(p/2+\varepsilon)^2}{n} \tag{2}$$

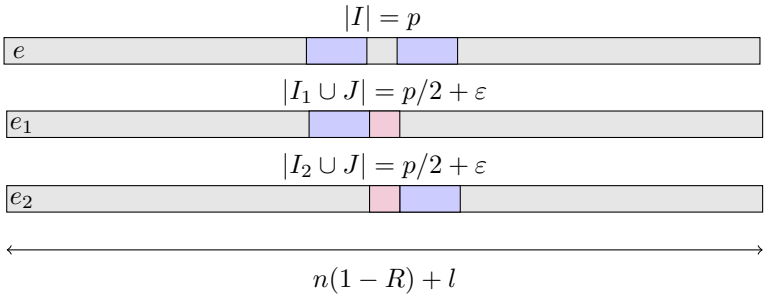
Rajouter dans le poids un ε défini de cette façon peut soulever une interrogation. L'origine de cette idée vient de la proposition suivante (dont on trouvera une preuve en annexe F)

Proposition 5. *Soient* e_1, e_2 *deux vecteurs aléatoires de poids* $p/2+\varepsilon$ *où* ε *est défini comme précédemment. Alors* $e_1 + e_2$ *aura un poids moyen égal à* p*.*

Dans l'algorithme de MMT on va construire comme dans l'algorithme de Dumer des vecteurs e_1, e_2 de poids $p/2 + \varepsilon$ sur lesquels on va opérer une collision que l'on précisera et on renverra le vecteur $e_1 + e_2$. D'après la proposition qui précède, on peut associer aux e_i un ensemble I_i de taille $\frac{p}{2}$ tels que $I_1 \cap I_2 = \emptyset$ et un ensemble J représentant l'intersection de leur support ($|J| = \varepsilon$). La figure 5 illustre ce choix d'ensemble. On appelle le triplet (I_1, I_2, J) une représentation du vecteur $e_1 + e_2$. Avec cette nouvelle méthode il n'y a plus « unicité » des solutions. Intuitivement plusieurs sous-solutions donneront la même solution. Par exemple si $p = 2$ et $k + l = 5$, les couples $(\{1,3\},\{2,5\},\{4\})$ et $(\{1,5\},\{2,3\},\{4\})$ donneront les mêmes solutions. Ainsi on montre qu'une même solution a de multiples représentations :

$$N_{MMT} = \binom{p}{p/2} \cdot \binom{n(1-R)+l-p}{\varepsilon}$$

FIGURE 5 – Décomposition d'un ensemble de représentation



Asymptotiquement,

$$\log_2(N_{MMT}) \sim p + (n(1-R)+l-p) \cdot h\left(\frac{\varepsilon}{n(1-R)+l-p}\right)$$

L'idée pour éviter l'écueil de plusieurs représentations menant à une même solution consiste à diviser la taille du syndrome cible et traiter encore deux sous-problèmes distincts. En effet le fait d'avoir plusieurs représentations menant à une même solution posera problème car nous aurons des « doublons » dans les solutions. Il deviendra alors difficile de compter le nombre de solutions obtenues.

Nous allons commencer par présenter cette idée de façon informelle puis montrer comment choisir les paramètres afin d'éviter des solution multiples. Nous terminerons ensuite par la présentation formelle de l'algorithme MMT ainsi que l'étude de sa complexité.

On commence par diviser la matrice G_2 et le syndrome cible en deux parties :

$$G_2 = \begin{bmatrix} A^1 \\ A^2 \end{bmatrix} \text{ où } A^1 \in \mathbb{F}_2^{(l-r)\times(n(1-R)+l)} \text{ et } A^2 \in \mathbb{F}_2^{r\times(n(1-R)+l)} \quad ; \quad 0_{[l]} = \begin{pmatrix} 0_{[l-r]\}} \\ 0_{[r]} \end{pmatrix}$$

où $0 \leq r \leq l$ est un nouveau paramètre. Ce degré de liberté supplémentaire va être utilisé comme « filtre » sur les représentations.

On commence par construire avec **Dumer** (technique de fusion) la liste :

$$\mathcal{L}(r,\varepsilon) = \{(e, G_2 e^t) \in \mathbb{F}_2^l \times \mathbb{F}_2^{n(1-R)+l} \mid (G_2 e)_{[r]} = A^2 e^t = 0_{[r]} \text{ et } w_H(e) = p/2 + \varepsilon\}$$

On merge ensuite la liste sur elle même (la fusion sur un même vecteur e n'est pas autorisée) par rapport à la seconde coordonnée pour les $l - r$ premiers bits, ce qui donne un ensemble :

$$\mathcal{L} = \{(e_1, e_2) \in \mathcal{L}(r,\varepsilon) \ : \ e_1 \neq e_2 \text{ et } A^1 e_1 = A^1 e_2\}$$

On renvoie alors l'ensemble des $e_1 + e_2$. Il est clair que le poids moyen de ces vecteurs est p et que $G_2(e_1 + e_2) = 0_{[l]}$.

La paramètre r va ici nous permettre de n'avoir en moyenne qu'une représentation menant à une solution. Intuitivement plusieurs représentations d'une même solution $e_1 + e_2$ ne permettront pas de passer le test $A^2 e_1^t = A^2 e_2^t = 0_{[r]}$. Plus formellement quelle paramètre r doit-on choisir afin de ne garder en moyenne qu'une représentation d'une solution finale ? C'est l'objet de la proposition qui suit dont on on trouvera une preuve en Annexe F

Proposition 6. *Supposons que $r \sim \log_2(N_{MMT})$. Alors pour toute solution e_s, une seule de ses représentations en moyenne sera dans les solutions finales.*

Dans toute la suite on imposera donc la contrainte :

$$r = p + (n(1-R) + l - p) \cdot h\left(\frac{\varepsilon}{n(1-R)+l-p}\right) \tag{3}$$

Remarque 10. *En offrant une plus grande flexibilité dans le choix des supports I_1 et I_2, donc plus de collisions, la technique des représentations nous permet déjà de gagner par rapport au paradoxe des anniversaires classique de Dumer. Malheureusement ce gain n'est que polynomial car la probabilité que les collisions se trouvent dans $[1, \cdots, (n(1-R)+l)/2]$, $[(n(1-R)+l)/2, n]$ est la plus élevée. Le véritable gain (dans l'exposant asymptotique) se cache derrière le fait de faire deux fusions.*

Nous allons maintenant rentrer dans les détails algorithmiques de l'amélioration de MMT faite sur la fusion de Dumer.

On utilisera le fait qu'asymptotiquement :

$$\binom{u/2}{v/2} = \tilde{O}\left(\sqrt{\binom{u}{v}}\right)$$

et les complexités données dans la sous-section qui précèdent pour obtenir la proposition qui suit :

Proposition 7 (Complexité de MMT(fusion))**.**
Le complexité de MMT(fusion) est

$$\tilde{O}\left(\sqrt{\binom{n(1-R)+l}{p/2+\varepsilon}} + \frac{\binom{n(1-R)+l}{p/2+\varepsilon}}{2^r} + \frac{\binom{n(1-R)+l}{p/2+\varepsilon}^2}{2^{2r+l-r}}\right)$$

et il fournit un nombre d'équations

$$\frac{\binom{n(1-R)+l}{p/2+\varepsilon}^2}{2^{r+l}}$$

MMT(fusion) (amélioration du Dumer Fusion)

Input : $A^1 \in \mathbb{F}_2^{(l-r)\times(n(1-R)+l)}$, $A^2 \in \mathbb{F}_2^{r\times(n(1-R)+l)}$, p
Output : $\mathcal{S}$ /* Ensemble de vecteurs résolvant le Dumer Fusion*/
$A^2 = \left[A_1^2|A_2^2\right]$ où $A_i^2 \in \mathbb{F}_2^{r\times(n(1-R)+l)/2}$
$\mathcal{S} \leftarrow [\,]$ (initialisation de liste)
$\mathcal{T} \leftarrow [\,]$ (initialisation d'une table hachage)
$\mathcal{I} \leftarrow$ **Dumer**$(A_1^2, A_2^2, p/2+\varepsilon)$ /* Résolution d'une sous fusion par Dumer */
for $e = (e_1, e_2) \in \mathcal{I}$ **do**
    $x \leftarrow A^1 e^t$ /* Permettra de faire le test sur les $l-r$ premiers bits */
    $\mathcal{T}[x] \leftarrow \mathcal{T}[x] \cup \{e\}$
end for
for $e \in \mathcal{I}$ **do**
    $x \leftarrow A_1 e_2^t$
    for $f \in \mathcal{T}[x]$ **do**
        if $e \neq f$ **then**
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{(e+f)\}$
        end if
    end for
end for

Il nous faut maintenant trouver les paramètres p, l, r, ε pour que l'amélioration de MMT satisfasse les contraintes de notre problème (temps amorti 1 et nombre d'équations inférieur à $P_{p+Rn/2(1+o(1))}$ du décodage statistique). La définition de ε donnée par l'équation (2) est équivalente à :

$$\varepsilon = \frac{1-p-\sqrt{1-2p}}{2} := f(p)$$

De plus comme nous l'avons remarqué en (3), r n'est pas un paramètre mais une fonction g de p et l :

$$g(p,l) = p + (n(1-R)+l-p)\cdot h\left(\frac{\varepsilon}{n(1-R)+l-p}\right)$$

Notre algorithme n'a donc que deux paramètres p et l. On note

$$L = \binom{n(1-R)+l}{p/2+\varepsilon}$$

Afin d'avoir un temps amorti de 1, L doit vérifier asymptotiquement :

$$\frac{L^2}{2^l} = \max\left(L, 2^r\sqrt{L}\right)$$

ce qui revient à choisir l tel que :

$$l = 2(n(1-R)+l)h\left(\frac{p/2+\varepsilon}{n(1-R)+l}\right)$$
$$-\max\left((n(1-R)+l)h\left(\frac{p/2+\varepsilon}{n(1-R)+l}\right)\ ,\ g(p,l)+\frac{1}{2}\cdot(n(1-R)+l)h\left(\frac{p/2+\varepsilon}{n(1-R)+l}\right)\right)$$

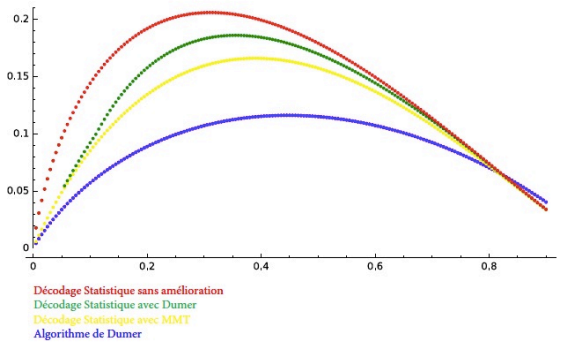
On va donc choisir un p, en déduire ε et trouver le l qui satisfait l'équation qui précède. Il faut ensuite vérifier que :

$$\log_2\left(\frac{L^2}{2^l}\right) \leq -2\cdot(p+\frac{Rn-l}{2})\cdot\log_2(1-2\tau_{GV})$$

Tout comme pour Dumer, nous normalisons ces équations. J'ai donc écrit un programme en Sage (cf annexe E) qui minimise la fonction $\pi \mapsto \pi + \frac{R-\lambda}{2}$. On obtient alors les courbes 4 et 5. Nous en déduisons

que MMT apporte une amélioration non négligeable, en particulier pour les rendement proche de 0.8 où l'approche de MMT permet de gagner bien plus que celle de Dumer. Cependant on remarque qu'en 0 il ne semble pas y avoir d'amélioration par rapport à Dumer. Un développement limité nous confirme que l'exposant au voisinage de 0 pour MMT(fusion) est le même que pour Dumer(fusion).

FIGURE 6 – Comparaison des exposants asymptotiques



3.3 L'amélioration de MMT à deux niveaux

Nous présentons ici une amélioration de MMT. Celle-ci consiste à introduire deux niveaux de représentations au lieu d'un. De façon informelle, nous allons remplacer dans MMT l'algorithme de Dumer par MMT.

Dans l'algorithme MMT nous avions construit la liste

$$\mathcal{L}(r,\varepsilon) = \{(e, G_2 e^t) \in \mathbb{F}_2^{Rn+l} \mid (G_2 e)_{[r]} = A^2 e^t = 0_{[r]} \text{ et } w_H(e) = p/2 + \varepsilon\}$$

avec une fusion de type Dumer pour ε et r choisis de façon à avoir un unique représentant par solution finale. L'idée ici consiste à ne plus construire cette liste avec Dumer mais avec MMT. Nous allons donc rajouter un niveau de représentation tout en gardant les mêmes paramètres r et ε. Comme dans l'algorithme MMT nous découpons la matrice A^2 en deux :

$$A^2 = \begin{bmatrix} X^1 \\ X^2 \end{bmatrix} \text{ où } X^1 \in \mathbb{F}_2^{(r-r_1)\times(n(1-R)+l)} \text{ et } X^2 \in \mathbb{F}_2^{r_1\times(n(1-R)+l)}$$

où $0 \leq r_1 \leq r$ est un nouveau paramètre. Comme dans ce qui précède on va choisir r_1 de sorte à filtrer les différentes représentations.

On construit avec **Dumer** la liste :

$$\mathcal{L}_1(r_1,\varepsilon_1) = \{(e, A^2 e^t) \mid (A^2 e^t)_{[r_1]} = X^2 e^t = 0_{[r_1]} \text{ et } w_H(e) = p/4 + \varepsilon_1\}$$

L'objectif ici est de choisir ε_1 de façon avoir des solutions de poids final $p/2+\varepsilon$ et ensuite de choisir r_1 pour filtrer toutes les représentations menant à une solution de $\mathcal{L}(r,\varepsilon)$. D'après la proposition 3, on choisit ε_1 tel que :

$$\varepsilon_1 = \frac{\varepsilon}{2} + \left(\frac{p}{4} + \varepsilon_1\right)^2$$

Le nombre de représentations de ce niveau est alors :

$$\binom{p/2}{p/4} \cdot \binom{n(1-R)+l-p/2-\varepsilon}{\varepsilon_1}$$

On en déduit (cf raisonnement de MMT) que :

$$r_1 = \frac{p}{2} + (n(1-R)+l-\frac{p}{2}-\varepsilon)h\left(\frac{\varepsilon_1}{n(1-R)+l-\frac{p}{2}-\varepsilon_1}\right)$$

L'appel à l'algorithme **Dumer** coûte :

$$\sqrt{\binom{n(1-R)+l}{p/4+\varepsilon_1}} + \frac{\binom{n(1-R)+l}{p/4+\varepsilon_1}}{2^{r_1}}$$

et fournit

$$\frac{\binom{n(1-R)+l}{p/4+\varepsilon_1}}{2^{r_1}}$$

solutions. La fusion qui suit fournit alors

$$\frac{\binom{n(1-R)+l}{p/4+\varepsilon_1}^2}{2^{r_1+r}}$$

solutions. L'amélioration MMT à deux niveaux consiste à utiliser cet algorithme à la place de Dumer dans `MMT` à la ligne 5. Si on reprend les calculs de complexité, on obtient la proposition qui suit :

Proposition 8 (Complexité de MMT(fusion) à deux niveaux)**.** *Le complexité de MMT(fusion) à deux niveaux est*

$$\tilde{O}\left(\sqrt{\binom{n(1-R)+l}{p/4+\varepsilon_1}} + \frac{\binom{n(1-R)+l}{p/4+\varepsilon_1}}{2^{r_1}} + \frac{\binom{n(1-R)+l}{p/4+\varepsilon_1}^2}{2^{r_1+r}} + \frac{\binom{n(1-R)+l}{p/4+\varepsilon_1}^4}{2^{2(r_1+r)+l-r}}\right)$$

et il fournit un nombre d'équations de parité :

$$\frac{\binom{n(1-R)+l}{p/4+\varepsilon_1}^4}{2^{2r_1+r+l}}$$

Pour satisfaire les contraintes que nous nous étions imposés on procède exactement de la même façon que dans l'amélioration de MMT. On commence par calculer ε, r puis on calcule ε_1 comme fonction de p et ε, donc fonction de p. On remarque ensuite que r_1 est une fonction de p, l et R. On vérifie que le r_1 obtenu est bien inférieur à r. Pour le coup amorti on a cette fois une résolution à faire légèrement plus compliquée avec un max portant sur trois valeurs. On écrit un programme déterminant le l en fonction de p et on fait ensuite exactement la même chose que pour MMT.

Malheureusement le fait de rajouter un niveau ne nous permet pas de gagner dans l'exposant aysmptotique, nous perdons même légèrement. En revanche, la description que nous avons faite de deux niveaux de représentation nous permettra de bien comprendre l'algorithme que nous présentons dans la sous-section suivante.

3.4 L'amélioration de BJMM

Nous présentons ici *rapidement* la dernière amélioration, inspirée de l'algorithme de décodage générique de Becker, Joux, May et Meureur (que l'on abréviera par BJMM). L'idée de l'algorithme de BJMM est de « pousser plus loin » la technique des représentations de l'algorithme MMT à deux niveaux.

Dans l'algorithme de MMT à deux niveaux nous avions choisi ε_1 et ε de telle façon à obtenir des mots de poids moyen $p/2+\varepsilon$ et p. L'idée de BJMM consiste à laisser « grossir » ε et ε_1. De cette façon on obtient plus de représentations d'une solution mais nous obtenons des mots de poids distinct de $p/2+\varepsilon$ et p lors de la fusion des listes $\mathcal{L}_1(r_1, \varepsilon_1)$ et $\mathcal{L}(r, \varepsilon)$. Dans BJMM, lors de la fusion des listes on ne va garder que les mots de bon poids, ce qui oblige une vérification contrairement à MMT. Nous devrons donc tenir compte des probabilités d'obtenir les bons poids. En revanche, nous gardons ici les mêmes expressions pour les paramètres r_1 et r (fonctions entre autre de ε et ε_1) de façon à n'obtenir qu'une représentation en moyenne par solution.

Il est clair que la somme de deux mots de poids $p/4+\varepsilon_1$ donne un mot de poids $p/2+\varepsilon$ avec la probabilité :

$$\alpha_1 = \frac{\binom{p/4+\varepsilon_1}{\varepsilon_1-\varepsilon/2} \cdot \binom{n(1-R)+l-p/4-\varepsilon_1}{p/4+\varepsilon/2}}{\binom{n(1-R)+l}{p/4+\varepsilon_1}}$$

Pour l'autre cas nous avons une probabilité de :

$$\alpha = \frac{\binom{p/2+\varepsilon}{\varepsilon} \cdot \binom{n(1-R)+l-p/2-\varepsilon}{p/2}}{\binom{n(1-R)+l}{p/2+\varepsilon}}$$

On en déduit alors la proposition qui suit.

Proposition 9 (Complexité de BJMM(fusion))**.** *La complexité de BJMM(fusion) est*

$$\tilde{O}\left(\sqrt{\binom{n(1-R)+l}{p/4+\varepsilon_1}} + \frac{\binom{n(1-R)+l}{p/4+\varepsilon_1}}{2^{r_1}} + \frac{\binom{n(1-R)+l}{p/4+\varepsilon_1}^2}{2^{r_1+r}} + \alpha_1^2 \cdot \frac{\binom{n(1-R)+l}{p/4+\varepsilon_1}^4}{2^{2(r_1+r)+l-r}}\right)$$

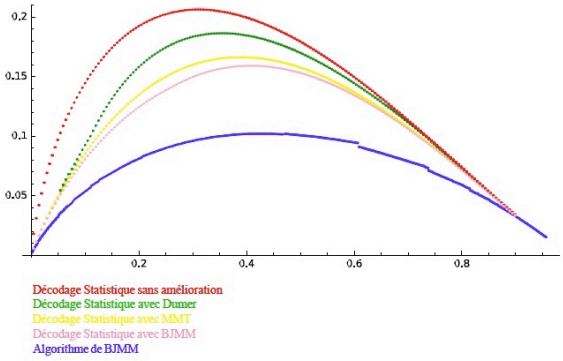
et il fournit un nombre d'équations de parité :

$$\alpha \cdot \alpha_1^2 \cdot \frac{\binom{n(1-R)+l}{p/4+\varepsilon_1}^4}{2^{2r_1+r+l}}$$

Pour notre problème de minimisation dans BJMM, ε et ε_1 ne sont plus des valeurs fixées mais des paramètres à optimiser. Nous avons fait le choix (à ε_1 et ε fixés) d'avoir la même contrainte sur l que précédemment. Le problème de ce choix est que désormais les solutions ne sont plus obtenues en temps amorti 1 et il faudra en tenir compte dans le calcul de la complexité finale de l'algorithme (le temps amortie est de $\frac{1}{\alpha}$).

Dans les figures 7 et 8 nous comparons les exposants asymptotiques du décodage statistique avec l'algorithme de décodage ISD de BJMM (l'un des plus efficaces actuellement). Nous avons encore une amélioration avec BJMM de l'exposant asymptotique du décodage statistique. Toutes les améliorations que nous avons présenté, inspirées des algorithmes ISD, ont donc permis un gain significatif, particulièrement pour les rendements compris entre 0.3 et 0.5.

FIGURE 7 – Comparaison des exposants asymptotiques



En résumé, nous avons dans l'étude du décodage statistique, dans un régime d'erreur sur la borne de Gilbert-Varshamov, cherché à baisser le poids des équations de parité, ce qui a permis des améliorations significatives. Nous avons actuellement d'autres pistes pour améliorer le décodage statistique, s'inspirant de l'idée de Dumer consistant à regarder le code poinçonné. Le début de ma thèse avec Jean-Pierre Tillich sera consacré à l'étude de ces différentes idées ainsi que la rédaction d'un article présentant les résultats que nous venons d'exposer.

Annexe A

Entropie et binomiales

Dans cette annexe nous rappelons la définition de l'entropie ainsi que différentes inégalités la reliant aux binomiales.

Définition 1 (Entropie)**.**
La fonction entropie binaire h est définie sur $[0,1]$ comme :

$$h(x) = -x\log_2(x) - (1-x)\log_2(1-x)$$

avec $h(0) = h(1) = 0$.

Proposition 1 (Equivalent asymptotique des binomiales)**.**
Pour un ratio w/n fixé, on a

$$\binom{n}{w} \underset{n\to+\infty}{\sim} \frac{2^{n\cdot h\left(\frac{w}{n}\right)}}{\sqrt{2\pi w(1-w/n)}}$$

Cette équivalent est précis, même pour $0 < w < n$,

$$\frac{2^{n\cdot h\left(\frac{w}{n}\right)}}{\sqrt{8w(1-w/n)}} \leq \binom{n}{w} \leq \frac{2^{n\cdot h\left(\frac{w}{n}\right)}}{\sqrt{2\pi w(1-w/n)}}$$

En particulier, on a :

$$\binom{n}{w} = \tilde{O}\left(2^{n\cdot h\left(\frac{w}{n}\right)}\right)$$

ce qui nous permet d'affirmer que :

$$\lim_{n\to+\infty} \frac{1}{n}\log_2\binom{n}{w} = h\left(\frac{w}{n}\right)$$

Proposition 2 (Somme de binomiales)**.**
Pour tout $0 < w < n/2$, on a

$$\frac{2^{n\cdot h\left(\frac{w}{n}\right)}}{\sqrt{8w(1-w/n)}} \leq \sum_{i=0}^{w}\binom{n}{i} \leq 2^{n\cdot h\left(\frac{w}{n}\right)}$$

Nous avons utilisé le résultat qui suit dans deux nombreux calculs de complexité :

Proposition 3.

$$\binom{n/2}{w/2}^2 = \frac{\binom{n}{w}}{\sqrt{\pi w/2(1-w/n)}}$$

En particulier,

$$\binom{n/2}{w/2} = \tilde{O}\left(\sqrt{\binom{n}{w}}\right)$$

Annexe B

Codes correcteurs

L'objectif de cette annexe est de présenter les bases de la théorie des codes correcteurs. Nous ne présenterons pas les codes dits structurés car ce n'est pas l'objet de ce rapport. Nous n'insisterons pas non plus sur le lien entre codes correcteurs et théorie de l'information.

B.1 Codes correcteurs

B.1.1 Premières définitions

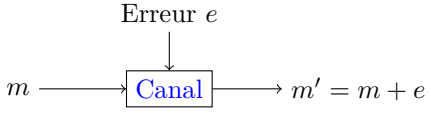
Définition 2 (Code Correcteur)**.**

Soit $n \in \mathbb{N}^*$. Un code correcteur binaire linéaire $\mathcal{C}$ est un sous-espace vectoriel de $\mathbb{F}_2^n$. On appelle l'entier n la longueur du code. La dimension de $\mathcal{C}$ est sa dimension en tant que $\mathbb{F}_2$-espace vectoriel et est en général notée k :

$$k = \dim_{\mathbb{F}_2} \mathcal{C}.$$

Le rapport $\frac{k}{n}$ est appelé rendement du code $\mathcal{C}$. Dans toute la suite, on notera R pour désigner le rendement.

L'objectif initial des codes correcteurs fut de permettre une transmission de l'information sans perte ou dit autrement, une fois un message reçu être en mesure de retrouver le message transmis. On modélise cela par le schéma suivant :



Le mot de code $m \in \mathcal{C}$ est transmis à travers *un canal de transmission* susceptible de produire des erreurs. A la sortie, on obtient au lieu de m, un mot m', et il s'agit de reconstituer m dans la mesure du possible. On appelle ce problème de façon informelle, le problème du décodage. Nous allons présenter dans B.1.3 un modèle de canal de transmission : le canal binaire symétrique. Il existe bien d'autres modèles de canaux (à effacement, gaussien, etc...) mais nous nous restreindrons à ce modèle car c'est le seul que nous avons considéré dans le rapport.

On rappelle que le poids de Hamming d'un vecteur $x = (x_1, \cdots, x_n) \in \mathbb{F}_2^n$ est défini comme :

$$w_H(x) = \#\{i \in \{1, \cdots, n\} \mid x_i \neq 0\}$$

et la distance de Hamming entre deux vecteurs x et y :

$$d_H(x, y) = w_H(x - y)$$

Il est facile de démontrer que d_H est une distance sur $\mathbb{F}_2^n$. On notera donc $B_H(c, r)$ une boule de centre c et de rayon r pour cette métrique.

Définition 3 (Distance minimale)**.**
La distance minimale d'un code $\mathcal{C}$ de longueur n est définie comme :

$$d_{\min}(\mathcal{C}) = \min_{x,y\in\mathcal{C},\ x\neq y}\{d_H(x,y)\}$$

Par linéarité, on peut la définir de façon équivalente comme :

$$d_{\min}(\mathcal{C}) = \min_{x\in\mathcal{C},\ x\neq 0}\{w_H(x)\}$$

Remarque 1. *Dans toute la suite on dira que $\mathcal{C}$ est un $[n,k,d]-$code (ou tout simplement code) s'il s'agit d'un code correcteur binaire linéaire de longueur n, de dimension k et de distance minimale d.*

La distance minimale va quantifier la capacité théorique de correction du code. En particulier, le lemme qui suit nous permet d'affirmer, que pour un code de distance minimale d nous pourrons corriger un mot ayant moins de $\lfloor\frac{d-1}{2}\rfloor$ erreurs.

Lemme 1. *Soit $\mathcal{C}$ un $[n,k,d]-$code. Alors les boules de centre $c\in\mathcal{C}$ de rayon $\lfloor\frac{d-1}{2}\rfloor$ sont deux à deux disjointes, autrement dit :*

$$\forall c,c'\in\mathcal{C},\ B_H\left(c,\lfloor\frac{d-1}{2}\rfloor\right)\cap B_H\left(c',\lfloor\frac{d-1}{2}\rfloor\right)=\emptyset$$

Démonstration. Ce lemme découle de l'inégalité triangulaire.

□

Reprenons les notations de ce qui précède et supposons que $m'=m+e$. L'objectif ici est de retrouver m ou de façon équivalente e. Il est clair que $m'\in B_H(m,w_H(e))$. L'idée naturelle pour retrouver m serait donc tester pour tous les $c\in\mathcal{C}$ l'appartenance de m' à la boule $B_H(c,w_H(e))$. Le problème étant que m' pourrait très bien appartenir à plusieurs de ces boules et que $w_H(e)$ n'est pas connu. Laquelle doit-on choisir ? Dans le cas où $w_H(e)\leq\lfloor\frac{d-1}{2}\rfloor$, le lemme qui précède nous apprend qu'il y a une unique telle boule et on peut donc conclure en regardant toutes les boules $B_H\left(c,\lfloor\frac{d-1}{2}\rfloor\right)$.

B.1.2 Comment décrire un code ?

Il y a deux façons usuelles de représenter un code correcteur qui sont deux manière naturelles de décrire un sous-espace vectoriel de $\mathbb{F}_2^n$. La première est donnée comme l'image d'une matrice et l'autre comme un noyau.

Définition 4 (Matrice génératrice)**.**
Soit $\mathcal{C}$ un $[n,k,d]-$code. Une matrice génératrice de $\mathcal{C}$ est une matrice $G\in\mathbb{F}_2^{k\times n}$ dont les lignes forment une base de $\mathcal{C}$. Autrement dit,

$$\mathcal{C}=\{mG\mid m\in\mathbb{F}_2^k\}$$

Définition 5 (Matrice de parité)**.**
Soit $\mathcal{C}$ un $[n,k,d]-$code. Une matrice de parité de $\mathcal{C}$ est une matrice $H\in\mathbb{F}_2^{(n-k)\times n}$ telle que

$$\mathcal{C}=\{x\in\mathbb{F}_2^n\mid xH^t=0\}$$

Remarque 2. *On parle de matrice génératrice et de parité car ces dernières ne sont pas uniques pour un code fixé.*

Si on dispose de la matrice génératrice d'un code $\mathcal{C}$, l'opération de « codage » d'un message $m\in\mathbb{F}_2^k$ que nous avons évoqué précédemment consistera uniquement en l'opération mG.

Remarque 3. *Une matrice de parité d'un code $\mathcal{C}$ peut être vue comme la matrice génératrice du code orthogonal $\mathcal{C}^\perp$. En théorie des codes on appelle le code $\mathcal{C}^\perp$, code dual de $\mathcal{C}$.*

Cette description des codes correcteurs nous permet de définir la notion d'équivalence de code.

Définition 6 (Codes équivalents)**.**
Deux codes $\mathcal{C}$ et $\mathcal{C}'$ sont dits équivalents si deux de leurs matrices génératrices sont équivalentes ou de façon équivalente pour leur matrice de parité. Autrement dit on peut passer de l'un à l'autre en permutant les coordonnées.

Remarque 4. *Deux codes équivalents n'ont aucune raison d'être identiques.*

Supposons que l'on dispose d'une matrice génératrice G d'un code $\mathcal{C}$. Alors toute opération de permutation sur les lignes donnera toujours une matrice génératrice de $\mathcal{C}$. En effet, permuter les lignes ne fait que changer l'ordre des vecteurs de la bases. En revanche, une opération de permutation des colonnes donnera une matrice génératrice d'un code équivalent.

Définition 7 (Forme systématique)**.**
On dit que la matrice génératrice d'une $[n, k, d]$-code est sous forme systématique si elle est de la forme :

$$[I_k|G_1] \text{ où } G_1 \in \mathbb{F}_2^{k\times(n-k)}$$

et pour une matrice de parité si elle est de la forme :

$$[I_{n-k}|H_1] \text{ où } H_1 \in \mathbb{F}_2^{(n-k)\times k}$$

La deuxième partie de la forme systématique d'une matrice génératrice ou de parité d'un code est appelée matrice de redondance. En effet, si l'on code un message $m \in \mathbb{F}_2^k$ dans un code de matrice génératrice $[I_k|G_1]$, on aura $(m|mG_1)$.

Une forme systématique pour une matrice d'un code a un autre avantage. Elle permet à partir d'une matrice génératrice sous cette forme de calculer une matrice de parité et réciproquement. En effet si $[I_k|G_1]$ engendre $\mathcal{C}$, alors il est facile de vérifier que :

$$\left[I_{n-k} \mid G_1^t\right]$$

est une matrice de parité de $\mathcal{C}$ et réciproquement.

Remarque 5. *La caractéristique 2 est ici primordiale.*

Si $G \in \mathbb{F}_2^{k\times n}$ engendre un $[n, k, d]-$code $\mathcal{C}$, on peut par un pivot de Gauss mettre G sous la forme : $[I_k|G_1]$. On peut alors être tenté de dire que cette matrice est une matrice génératrice sous forme systématique de $\mathcal{C}$. Ceci est faux. En effet, dans le pivot de Gauss on a fait des permutation sur les lignes et sur les colonnes. On obtient donc la matrice génératrice d'un code équivalent à $\mathcal{C}$. C'est pour cette raison qu'à partir de maintenant nous n'autoriserons pas les permutations sur les colonnes dans les pivots de Gauss.

Soit I un sous-ensemble de $\{1, \cdots, n\}$ de taille $|I|$, on utilisera la notation suivante :
· Si $[A|B] \in \mathcal{M}_{k,n}(\mathbb{F}_2)$ où $A \in \mathcal{M}_{k,|I|}(\mathbb{F}_2)$ et $B \in \mathcal{M}_{k,n-|I|}(\mathbb{F}_2)$, l'écriture $M = [A|B]_I$ signifiera que $M_I = A$ et $M_{\{1,\cdots n\}-I} = B$;

De cette façon avec nos pivots de Gauss, si $G \in \mathbb{F}_2^{k\times n}$ est la matrice génératrice d'un $[n, k, d]-$code, on peut obtenir avec nos pivots une matrice de la forme :

$$[I_k \mid G_1]_I$$

pour tout I un sous-ensemble de $\{1 \cdots, n\}$ de taille k tel que G_I est inversible. Dans la section B.2, nous démontrons que si l'on tire un ensemble I de taille k de façon uniforme parmi $\mathcal{P}(\{1, \cdots, n\})$, alors G_I n'est pas inversible avec une probabilité exponentiellement faible.

Dans la parie principale du rapport, lorsque nous évoquions des pivots de Gauss, nous faisions implicitement le raisonnement suivant : on tire un ensemble I comme précédemment au hasard, avec un pivot de Gauss sans permutation des colonnes on met G sous la forme $[I_k \mid G_1]_I$. Si nous échouons nous tirons

alors un nouvel ensemble et on recommence. Cet algorithme est polynomial car un pivot de Gauss classique se fait en temps polynomial et la probabilité de tirer un bon ensemble est exponentiellement proche de 1 en n.

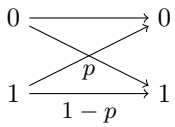
Nous sommes maintenant en mesure d'expliquer comment passer en temps polynomial d'une matrice génératrice G à une matrice de parité H. On met G sous la forme qui précède, H s'obtient alors comme :

$$[I_{n-k} \mid G_1]_I$$

Comme nous venons de l'expliquer, ceci a un coût polynomial.

B.1.3 Modèle du canal binaire symétrique

Le premier, et le plus classique, modèle de canal dans la théorie des codes est le canal binaire symétrique que l'on note $BSC(p)$ pour $p \in [0,1]$. Ce canal fonctionne de la façon suivante : si 0 est envoyé, alors le receveur reçoit 0 avec probabilité $1-p$ et 1 avec probabilité p et réciproquement si 1 est envoyé. On parle donc de canal symétrique car les modifications $0 \mapsto 1$ et $1 \mapsto 0$ ont même probabilité. De plus si n bits sont envoyé à travers le canal, on supposera que leurs comportements seront indépendants, on parle donc de canal sans mémoire. Ce canal est représenté dans le schéma qui suit.



La description rigoureuse du canal est : soit $x \in \mathcal{C}$ un vecteur que l'on transmet à travers le canal. On note $y = x + e$ le mot reçu. Alors $e = (e_1, \cdots, e_n)$ où les e_i sont des variables de Bernoulli de paramètre p et indépendantes. Dans toute la suite on dénotera ces hypothèses par $e \sim BSC(p)$.

Remarque 6. *On peut toujours supposer que $p \leq \frac{1}{2}$. En effet, si $p > \frac{1}{2}$, alors il suffit d'inverser tous les bits reçus pour se placer dans ce cas.*

Il est clair que si $e \sim BSC(p)$, alors :

$$\mathbb{E}\left(w_H(e)\right) = np \quad ; \quad Var\left(w_H(e)\right) = np(1-p)$$

On peut alors en déduire la proposition suivante :

Proposition 4 (Borne de Chernoff)**.**
Si $e \sim BSC(p)$, alors pour tout $\varepsilon > 0$,

$$\mathbb{P}\left(|w_H(e) - pn| \geq \varepsilon pn\right) \leq e^{-\frac{pn\varepsilon^2}{3}}$$

Cette proposition a une conséquence importante : *pour n tendant vers $+\infty$, les motifs d'erreurs du canal binaire symétrique auront un poids typique de pn.*

Il s'agit ici de préciser le terme de *typique*. Comme la série $\sum_n e^{-\frac{pn\varepsilon^2}{3}}$ converge pour tout $\varepsilon > 0$, un lemme classique (issu de Borell-Cantelli) nous permet d'affirmer que

$$\frac{w_H(e) - pn}{pn} \xrightarrow[n\to+\infty]{p.p.s} 0$$

où *p.p.s* dénote la convergence presque sûre. On en déduit donc que :

$$w_H(e) = pn(1 + u_n)$$

où u_n est une variable aléatoirement tendant presque sûrement vers 0. On écrira donc dans la suite :

$$w_H(e) = pn(1 + o(1))$$

Cette écriture nous permettra d'exprimer que pn est le facteur dominant. On peut aussi justifier cette écriture de façon plus simple. D'après l'inégalité de Tchebytchev :

$$\mathbb{P}\left(|w_H(e) - np| \geq Var(w_H(e))^{3/4}\right) \leq \frac{1}{\sqrt{Var(w_H(e))}}$$

Dans notre cas cette probabilité va tendre vers 0 pour $n \to +\infty$. De plus $Var(w_H(e))^{3/4}$ est négligeable devant np. On en déduit donc qu'asymptotiquement la probabilité que $w_H(e)$ soit plus grand qu'un facteur négligeable devant np est nulle. On peut donc supposer que pour n assez grand, $w_H(e)$ s'écrit bien $np(1 + o(1))$.

Nous allons maintenant revenir au problème du décodage dans le modèle du canal binaire symétrique. On démontre facilement que :

$$\mathbb{P}_{e\sim BSC(p)}(y|x) = \left(\frac{p}{1-p}\right)^{d_H(x,y)} \cdot (1-p)^n$$

Puisqu'on a supposé que $p < 1/2$, la probabilité précédente décroît lorsque i croît : *une erreur e est d'autant plus probable que son poids est plus faible.*

Si le récepteur reçoit y, une solution raisonnable au problème de décodage consisterait à choisir le vecteur x qui maximise la probabilité $\mathbb{P}_{e\sim BSC(p)}(y|x)$, ce qui est équivalent à trouver le vecteur $x \in \mathcal{C}$ le plus proche de y pour la distance de Hamming d_H. Or dans le cas où $w_H(e) \leq \lfloor \frac{d-1}{2} \rfloor$, le mot le plus proche de $\mathcal{C}$ est unique (cf lemme 1). On retrouver donc la définition de notre problème de décodage.

Cette définition du décodage comme mot le plus proche est appelée décodage par vraissemblance maximale.

B.2 Modèle aléatoire des codes correcteurs

Dans cette sous-section nous présentons le modèle des codes aléatoires. Intuitivement un code est dit aléatoire lorsque l'on tire sa matrice génératrice ou de parité de façon aléatoire. On s'intéresse ensuite aux questions comme quel est le poids moyen d'un mot du code ? Quelle est sa distance minimale en moyenne ?

B.2.1 Généralités

Définition 8 (Matrice aléatoire)**.**

Dans notre modèle, tirer une matrice aléatoire de $\mathbb{F}_2^{u\times v}$ est équivalent à considérer $u \times v$ variables de Bernoulli i.i.d de paramètre $1/2$.

Pour désigner cet évènement on écrira $M \sim \mathcal{A}(u, v)$

Remarque 7. *Chaque variable de Bernoulli représente une coordonnée de la matrice.*

Définition 9 (Générateurs de Elias)**.**

On fixe la longueur de bloc n, et le rendement $k = Rn$. Un code aléatoire de Elias $\mathcal{C}$ est un codé généré par une matrice $G \sim \mathcal{A}(k, n)$.

Pour désigner cet évènement on écrira $\mathcal{C} \sim \mathcal{G}(n, k)$.

Remarque 8. *Un code $\mathcal{C} \sim \mathcal{G}(n, k)$ est de dimension $\leq k$.*

Définition 10 (Générateurs de Gallager)**.**

On fixe la longueur de bloc n, et le rendement $k = Rn$. Un code aléatoire de Gallager $\mathcal{C}$ est un code de matrice de parité $H \sim \mathcal{A}(n - k, n)$.

Pour désigner cet évènement on écrira $\mathcal{C} \sim \mathcal{H}(n, k)$.

Remarque 9. *Un code $\mathcal{C} \sim \mathcal{G}(n, k)$ est de dimension $\geq k$.*

Ces deux procédés semblent « coïncider », cependant il faut les distinguer. Dans un cas la dimension du code obtenu est au plus (générateur) et dans l'autre au moins (parité). Ceci nous amène à penser que les résultats probabilistes vont être distincts.

Dans la suite on parlera de codes aléatoires générateurs si on se place dans le modèle $\mathcal{G}(n,k)$ et de codes aléatoires de parité dans le cas de $\mathcal{H}(n,k)$.

Comme nous venons de le remarquer, un code obtenu avec ces procédés n'est pas toujours de dimension k. La proposition qui suit nous montre cependant que la probabilité pour que la dimension ne soit pas k est exponentiellement faible en les paramètres.

Proposition 5.

$$\mathbb{P}_{\mathcal{C}\sim\mathcal{G}(k,n)}\left(\mathcal{C} \textit{ est de dimension } k\right) \geq 1 - \frac{1}{2^{n(1-R)}}$$

Démonstration. Par définition il est clair que

$$\mathbb{P}_{\mathcal{C}\sim\mathcal{G}(n,k)}\left(\mathcal{C} \text{ est de dimension } k\right) = \mathbb{P}_{G\sim\mathcal{A}(k,n)}\left(G : rg(G) = k\right)$$

Les probabilités se faisant sur des ensembles finis et de façon uniforme, nous allons chercher à calculer le nombre de matrices de rang k que l'on divisera ensuite par le nombre de matrice de $|\mathbb{F}_2^{k\times n}| = 2^{n\times k}$. Pour cela nous raisonnerons sur les lignes.

Pour la première ligne on a $2^n - 1$ possibilités (vecteurs non nuls). Une des ces possibilités étant fixée, pour la deuxième ligne il faut choisir un vecteur n'étant pas colinéaire à la première, c'est à dire $2^n - 2$ choix. Pour la troisième ligne on doit choisir un vecteur n'étant pas dans l'espace engendré par les deux premières, autrement dit on a $2^n - 2^2$ choix. En itérant le raisonnement on a donc :

$$\prod_{i=0}^{k-1}(2^n - 2^i) = 2^{nk}\prod_{i=0}^{k-1}(1 - 2^{i-n})$$

On en déduit que,

$$\mathbb{P}_G\left(rg(G) = k\right) = \prod_{i=0}^{k-1}(1 - 2^{i-n}) \geq 1 - 2^{k-n} = 1 - 2^{-n(1-R)}$$

la dernière égalité résultant du fait que $\prod_0^{k-1}(2^n - 2^i) \geq 2^n - 2^k$.

□

Remarque 10. *Dans le cas du modèle de Gallager il suffit de remplacer $1 - R$ par R dans la formule qui précède.*

Cette proposition nous permet d'affirmer que la probabilité qu'une matrice carré de $\mathbb{F}_2$ tirée de façon aléatoire ne soit pas inversible est exponentiellement faible. Dans l'annexe sur les algorithmes par ISD nous utiliserons ce résultat.

B.2.2 Distribution des mots vis à vis d'un code aléatoire

Nous intéressons maintenant à la probabilité qu'un mot de l'espace ambiant appartienne à un code aléatoire.

Proposition 6. *On se donne $y \in \mathbb{F}_q^n - \{0\}$, on a :*

$$\mathbb{P}_{\mathcal{C}\sim\mathcal{H}(n,k)}\left(y \in \mathcal{C}\right) = \frac{1}{2^{n-k}}$$

Démonstration. Par définition il est clair que

$$\mathbb{P}_{\mathcal{C}\sim\mathcal{H}(n,k)}\left(y \in \mathcal{C}\right) = \mathbb{P}_{H\sim\mathcal{A}(n-k,n)}\left(Hy^t = 0\right)$$

On note $H = (h_{i,j})_{i,j}$. Résoudre (en $h_{i,j}$) Hy^t revient à résoudre $n-k$ équations $(E_i) : \sum_{j=1}^{n} h_{i,j} y_j = 0$. Comme $y \neq 0$, quitte permuter les colonnes de la matrice et du vecteur, on peut supposer que $y_1 \neq 0$, autrement dit $y_1 = 1$. D'où :

$$\forall i \in \mathbb{N}, h_{i,1} = \sum_{j=2}^{n} h_{i,j} y_j$$

Les $h_{i,j}$ étant des variables de Bernoulli indépendantes de paramètre $1/2$, on obtient pour chaque E_i une probabilité $1/2$ de réussite, d'où le résultat.

□

On cherche maintenant à savoir combien un code aléatoire de parité a t-il en moyenne de mots de poids ω. On en déduira facilement combien il y a en moyenne de mots de poids $\leq w$

Pour un code $\mathcal{C}$, on note $A(\mathcal{C}, \omega)$ le nombre de mots de code de poids ω. Le choix des codes aléatoires se fera sur $\mathcal{H}(n,k)$. On peut alors voir $A(\cdot, \omega)$ comme une variable aléatoire de $\mathcal{H}(n,k) \to \mathbb{N}$.

Proposition 7.

$$\mathbb{E}_{\mathcal{C}\sim\mathcal{H}(n,k)}[A(\mathcal{C}, \omega = 0)] = 1$$

$$\mathbb{E}_{\mathcal{C}\sim\mathcal{H}(n,k)}[A(\mathcal{C}, \omega)] = \binom{n}{\omega} \frac{1}{2^{n-k}}, \quad \omega \geq 1$$

$$\mathbb{E}_{\mathcal{C}\sim\mathcal{H}(n,k)}[\#\{y \in \mathcal{C}, w_H(y) \leq \omega\}] = 1 + \sum_{k=1}^{w} \binom{n}{\omega} \frac{1}{2^{n-k}}$$

Démonstration. La premier point est clair, prouvons le second. Soit $p = \binom{n}{\omega}$, on note $\{y_1, \cdots, y_p\}$ l'ensemble des mots de poids ω. On introduit la variable aléatoire X_i sur $\mathcal{H}(n,k)$,

$$X_i(\mathcal{C}) = \begin{cases} 1 \text{ si } y_i \in \mathcal{C} \\ 0 \text{ sinon} \end{cases}$$

On a $A(\cdot, \omega) = \sum_{i=1}^{p} X_i$, donc par linéarité de l'espérance obtient

$$\mathbb{E}_{\mathcal{C}}[A(\mathcal{C}, \omega)] = \sum_{i=1}^{p} \mathbb{E}_{\mathcal{C}}[X_i(\mathcal{C})] = \sum_{i=1}^{p} \frac{1}{2^{n-k}}$$

ce qui donne le résultat. Le dernier point s'obtient tout simplement par linéarité de l'espérance.

□

Dans le modèle $\mathcal{G}(n,k)$ nous avons des résultatss similaires. Nous ne les présentons pas ici car dans la partie principale du rapport, nous nous intéressions aux équations de parité d'un code aléatoires.

Remarque 11. *Pour ω fixé,* $\lim_{n\to+\infty} \frac{1}{n} \log_2 \mathbb{E}_{\mathcal{C}\sim\mathcal{G}(n,k)} (A(\mathcal{C}, n\omega)) = h_2(\omega) + R - 1$

Nous allons maintenant étudier la répartition des mots de poids ω dans un code aléatoire.

Proposition 8. *On se donne $R \in [0,1[$ et $n \in \mathbb{N}$ tel que $Rn \in \mathbb{N}$. On s'intéresse à l'univers $\mathcal{H}(n, k = Rn)$.*
i) Pour tout $\delta \in]0, \frac{1}{2}[$:

$$\mathbb{P}\left(\sum_{w=1}^{n\delta} A(\mathcal{C}, w) \geq 1\right) \leq n\delta \cdot 2^{n(h_2(\delta)-(1-R))}$$

ii) Pour tout $\alpha > 0$, on a

$$\mathbb{P}_{\mathcal{C}}\left(\frac{A(\mathcal{C}, w)}{\mathbb{E}_{\mathcal{C}}[A(\mathcal{C}, w)]} \notin]1-\alpha, 1+\alpha[\right) \leq \frac{1}{\alpha^2 \mathbb{E}_{\mathcal{C}}\left(A(\mathcal{C}, w)\right)}$$

Démonstration. On rappelle que :

$$\binom{n}{w} \leq \frac{2^{nh(\frac{w}{n})}}{\sqrt{2\pi w(1-w/n)}}$$

On en déduit alors par l'inégalité de Markov et la proposition (?) le point i).
Le point ii) s'obtient par un calcul de variance et une application de l'inégalité de Tchébychev.

□

On rappelle que l'on note $\delta_{GV}(R)$ (borne de Gilbert-Varshamov) l'unique réel de $[0, \frac{1}{2}]$ tel que :

$$h(\delta_{GV}(R)) = 1 - R$$

Interprétation : Pour n assez grand le point i) nous apprend qu'il y a très peu de codes qui contiennent des mots de poids $< n\delta_{GV}$ ou de poids $> n(1 - \delta_{GV}(R))$. Ceci motive l'introduction de la fonction :

$$G(r, w) = \begin{cases} h_2(w) - (1 - R) \text{ si } \delta_{GV}(r) \leq w \leq 1 - \delta_{GV}(R) \\ 0 \text{ sinon} \end{cases}$$

On se fixe un rendement R. Pour $w/n \in [\delta_{GV}(R), 1 - \delta_{GV}(R)]$ le nombre de mot de codes de poids w pour un code aléatoire sur $\mathcal{H}(n, Rn)$ est de l'ordre de

$$2^{n(G(R,w/n)+o(1))}$$

D'après le point ii) les mots de code vont avoir tendance à se retrouver dans la différence de boules :

$$B(0, n(1 - \delta_{GV}(R))) - B(0, n\delta_{GV}(R))$$

Annexe C

Algorithmes de décodage par ensemble d'information

L'objectif de cette annexe est d'introduire à la cryptographie à base de codes correcteurs, de présenter le problème du décodage générique et les algorithmes de décodage dits par ensemble d'information pour le résoudre.

Nous avons vu dans la section 1 le crypto-système de McEliece. Cependant ici nous allons nous intéresser à sa variante *duale*, le crypto-système de Niederreiter qui comme nous le verrons lui est équivalent. Ce choix de présentation est motivé par le fait que les algorithmes de décodage par ensemble d'information ont été conçu pour la cryptanalyse du chiffrement à clef publique de Niedrerreiter.

Avant de présenter le crypto-système de Niederreiter, nous commencerons par introduire le concept de syndrome et quelques problèmes qui lui sont liés. Nous terminerons dans un troisième temps par la présentation des premiers algorithmes de décodage par ensemble d'information et leur généralisation.

C.1 La sécurité du calcul des syndromes comme fonction à sens unique

C.1.1 Préliminaires

Dans toute la suite $\mathcal{C}$ désignera un code binaire linéaire de paramètre $[n, k, d]$ défini par une matrice de parité $H \in \mathbb{F}_2^{(n-k)\times n}$. On notera $r = n - k$ la codimension du code.

Définition 11 (Syndrome)**.**
On appelle syndrome de $\mathcal{C}$, tout vecteur de la forme

$$yH^t \in \mathbb{F}_2^r \text{ où } y \in \mathbb{F}_2^n$$

On définit alors l'application de syndrome $\mathcal{S}_H$ relativement à H comme :

$$\begin{aligned} \mathcal{S}_H : \ & \mathbb{F}_2^n \to \mathbb{F}_2^r \\ & y \longmapsto yH^t \end{aligned}$$

Pour tout vecteur $s \in \mathbb{F}_2^r$, on notera l'ensemble des mots de $\mathbb{F}_2^n$ de syndrome s :

$$\mathcal{S}_H^{-1}(s) = \{y \in \mathbb{F}_2^n, \ S_H(y) = s\}$$

Par définition, on a $\mathcal{S}_H^{-1}(0) = \mathcal{C}$. L'ensemble $y+\mathcal{C}$ pour $y \in \mathbb{F}_2^n$ est appelé un coset de $\mathcal{C}$. Il est clair que la relation binaire d'appartenance à un coset est une relation d'équivalence. Un coset possédant 2^k éléments, on en déduit qu'il y a exactement 2^r cosets de $\mathcal{C}$. De plus pour une matrice de parité H de $\mathcal{C}$, on a une bijection entre les cosets et syndromes relativement à H.

Proposition 9. *Pour tout syndrome $s \in \mathbb{F}_2^r$, on a :*

$$\mathcal{S}_H^{-1}(s) = y + \mathcal{C}$$

où $y \in \mathbb{F}_2^n$ est n'importe quel mot de syndrome s. De plus, trouver un tel y à partir de s peut se faire en temps polynomial.

Démonstration. Pour tout $y, z \in \mathcal{S}_H^{-1}(s)$, on a $yH^t = zH^t$. Donc $(y+z)H^t = 0$, ce qui signifie que $y+z \in \mathcal{C}$ et il s'ensuit que $\mathcal{S}_H^{-1}(s) = y + \mathcal{C}$.

Pour calculer un vecteur de $\mathcal{S}_H^{-1}(s)$, on commence par mettre H sous forme systématique $H_0 = [I_r|X]$ à l'aide d'un pivot de Gauss.

Remarque 12. *On rappelle qu'une forme systématique est vu à permutation près des colonnes car dans les pivots de Gauss nous n'autorisons que les opérations sur les colonnes.*

Autrement dit, on a calculé une matrice $U \in \mathcal{G}L_r(\mathbb{F}_2)$ en temps $O(r^3)$ tel que $H_0 = UH$. On considère le vecteur $y = [sU^t|0] \in \mathbb{F}_2^n$ (vu à permutation près des colonnes).

$$yH^t = y(U^{-1}H_0)^t = yH_0^t(U^{-1})^t = (sU^t|0)\begin{bmatrix} I_r \\ X^t \end{bmatrix}(U^t)^{-1} = s$$

Les calculs matriciels se faisant en temps polynomial, on a le résultat. □

Remarque 13. *Il est à noter ici que le vecteur obtenu pour un syndrome donné a une forme particulière.*

C.1.2 Problèmes de décodage par syndrome

Etant donné un syndrome $s = yH^t \in \mathbb{F}_2^r$, le décodage par syndrome revient à résoudre l'un de ces trois problèmes équivalents :

(i) Trouver le vecteur $x \in \mathcal{C}$ le plus proche de y pour la distance de Hamming.
(ii) Trouver une erreur $e \in y + \mathcal{C}$ de poids de Hamming minimal.
(iii) Trouver une erreur $e \in \mathcal{S}_H^{-1}(s)$ de poids de Hamming minimal.

Dans la pratique, si on se fixe une instance du problème de décodage par syndrome, il est difficile de vérifier si le vecteur e est bien de poids minimal dans le coset. Intuitivement cela revient à vérifier en comparant avec tous les vecteurs. C'est pour cette raison que le problème du décodage par syndrome n'est pas dans NP. A la place, on considèrera un problème légèrement différent, le problème du décodage de syndrome à distance bornée :

Problème 1 (Décodage de syndrome à distance bornée)**.**
Entrée : $H \in \mathbb{F}_2^{r \times n}$, $s \in \mathbb{F}_2^r$, $w \in \mathbb{N}^*$;
Sortie : Un mot $e \in \mathcal{S}_H^{-1}(s)$ tel que $w_H(e) \leq w$.

On notera $CSD(H, w, s)$ une instance de ce problème.

Remarque 14. *Il est à noter que la version décisionnelle de ce problème, contrairement au problème du logarithme discret, lui est équivalente. Un attaquant peut résoudre toute instance de CSD avec un nombre linéaire d'appels à l'oracle de la version décisionnelle.*

Le fait d'ajouter w permet d'avoir un problème NP, la vérification consistant désormais en un calcul matriciel et de poids. Berlekamp, McEliece et Van Tiborg ont montré en 1978 dans [**?**] que la version décisionnelle de CSD est NP-complète.

La proposition qui suit montre que CSD est équivalent au problème de décodage que nous avions considéré dans la section 1. Nous commençons par rappeler le problème du décodage à distance bornée.

Problème 2 (Décodage à distance bornée)**.**
Entrée : $G \in \mathbb{F}_2^{k \times n}$, $x \in \mathbb{F}_2^n$, $w \in \mathbb{N}^*$;
Sortie : Un mot $e \in \mathbb{F}_2^n$ tel que $w_H(x + e) \leq w$.

Proposition 10. *Les problèmes 1 et 2 sont équivalents et ceci en temps polynomial (si on sait résoudre l'un, on peut résoudre l'autre avec une complexité égale à un facteur polynomial près).*

Démonstration. Soit Φ un algorithme permettant de résoudre le problème 1. Soit $x = mG + e$ la donnée d'un problème de décodage où $\omega_H(e) \leq \omega$. On calcule par un pivot de Gauss la matrice de parité H du code et on a

$$s = xH^t = eH^t$$

On utilise Φ qui permet alors de retrouver e. Le pivot de Gauss se faisant en temps polynomial on a le résultat.

Réciproquement, soit φ un algorithme permettant de résoudre le problème 2. Soit $s = eH^t$ une instance du problème 1. On tire aléatoirement un ensemble I de taille r jusqu'à ce que H_I soit inversible (H_I désigne la matrice H restreinte aux colonnes d'indice I). Notre discussion sur les codes aléatoires a montré qu'il suffit de faire un nombre de tirage polynomial pour trouver un tel I (la probabilité que I ne convienne pas est exponentiellement faible en n). Un syndrome étant de taille r, on peut calculer $s(H_I^{-1})^t$. Soit x le vecteur dont la restriction à I vaut ce vecteur et dont les autres coordonnées sont nulles. Dans le calcul qui suit on confondra l'ensemble I et $\{1, \cdots, r\}$.

$$xH^t = \sum_{j=1}^{n} x_j C_j(H) = \sum_{j \in I} x_j C_j(H)$$

Or pour $i \in I$,

$$x_i = \sum_{j \in I} s_j h_{i,j}^{-1}$$

Donc,

$$(xH^t)_i = \sum_{j \in I} \sum_{k \in I} h_{i,j} \cdot h_{j,k}^{-1} s_k = \sum_{k \in I} \left(H_I \cdot H_I^{-1}\right)_{i,k} s_k = s_i$$

D'où $xH^t = s$. On peut maintenant décoder x avec φ pour retrouver un vecteur e de poids inférieur à ω de même syndrome s. Ce vecteur est alors solution du problème 2. Nous n'avons fait que des opérations matricielles et un nombre de tirages au plus polynomial, d'où le résultat.

□

Remarque 15. *Cette preuve nous permet de comprendre pourquoi dans la suite nous dirons que l'on cherche à retrouver l'erreur pour résoudre CSD.*

Dans la théorie des codes correcteurs, tout comme en cryptographie, les problèmes 1 et 2 ont un intérêt si la solution est unique. En effet, dans le crypto-système de McEliece si la solution n'était pas unique le déchiffrement n'aurait aucun sens. De plus s'il y avait plusieurs solutions il serait a priori plus facile des les trouver. La cryptographie impose donc de trouver un w pour que les problèmes soient difficiles et qu'il n'y ait qu'une solution.

La théorie des codes correcteurs aléatoires nous permet de donner une valeur de w vérifiant ces deux contraintes : la borne de Gilbert-Varshamov. Cette dernière est asymptotiquement (en la longueur du code) à rendement fixé, donnée par :

$$t_{GV} = n \cdot h^{-1}(1 - R)$$

Ceci mérite quelques explications. L'ensemble des problèmes $CSD(H, w, s)$ pour des matrices H tirées de façon aléatoire donnera en moyenne :

$$\frac{\binom{n}{w}}{2^r}$$

solutions. Le plus grand nombre d'erreurs que l'on peut corriger, tout en gardant une solution en moyenne, est donc définie comme :

$$\binom{n}{w} = 2^r$$

Asymptotiquement cela correspond à $n \cdot h\left(\frac{w}{n}\right) = r$ où $r = n - k = n(1 - R)$. On retrouve donc la définition de t_{GV} comme le plus grand nombre d'erreurs tel qu'il y ait une solution en moyenne. De cette façon on « mesurera » l'efficacité des algorithmes permettant de résoudre les problèmes 1 et 2 en prenant $w = t_{GV}$ et de façon générale on se placera toujours dans le cas où $w < t_{GV}$.

C.2 La variante de Niederreiter

A la différence du crypto-système de McEliece où un clair est un mot de code, Niederreiter a proposé de le coder dans l'erreur avec une fonction :

$$\phi_{n,t} : \{0,1\}^l \to \mathcal{W}_{n,t}$$

où $\mathcal{W}_{n,t} = \{e \in \mathbb{F}_2^n,\ w_H(e) = t\}$ et $l = \log_2 \binom{n}{t}$. Le fonctionnement du système de chiffrement est résumé à la table C.1.

Clef secrète :
- $\mathcal{C}$ un code linéaire binaire $[n, k, d]$ choisi parmi une famille de code Γ représenté par une matrice de parité $H \in \mathbb{F}_2^{(n-k)\times n}$
- Un algorithme *efficace* de décodage de syndrome $\mathcal{D}$
- $M \in \mathcal{G}L_k(\mathbb{F}_2)$
- $P \in \mathcal{G}L_n(\mathbb{F}_2)$ matrice de permutation

Clef publique :
- $H' = MHP$
- t un entier inférieur à la capacité de correction du code
- $\phi_{n,t}$ qui permettra de représenter un clair par une erreur de poids t

Algorithme de chiffrement :
Texte chiffré s correspondant à un message clair m : $s = H'\phi_{n,t}(m)^t$

Algorithme de déchiffrement :
On calcule $M^{-1}s = HP\phi_{n,t}(m)^t$ puis on utilise l'algorithme de décodage de syndrome $\mathcal{D}$ pour retrouver $P\phi_{n,t}(m)^t$ que l'on multiplie ensuite par P^{-1} et que l'on inverse avec $\phi_{n,t}^{-1}$.

TABLE C.1 – Système de chiffrement à clef publique de Niederreiter

Tout comme dans le système de chiffrement de McEliece, la clef publique est une matrice de parité d'un code linéaire $\mathcal{C}'$ équivalent au code $\mathcal{C}$. Retrouver un clair à partir d'un chiffré peut encore une fois être fait de deux façons : retrouver la structure du code, donc l'algorithme de décodage, ou chercher un algorithme permettant de résoudre le problème 1 de façon efficace. On parle comme pour le crypto-système de McEliece d'attaque structurelle et de décodage générique. La proposition 10 montre que le décodage générique est équivalent de façon polynomiale pour les deux systèmes de chiffrement. Dans la suite on parlera donc de décodage générique indistinctement entre les chiffrements de McEliece et Niederreiter.

Dans la partie principale rapport nous avons étudié l'algorithme de décodage statistique qui est un algorithme de décodage générique pour le système de McEliece. Dans la prochaine section nous allons étudier les algorithmes de décodage génériques par ensemble d'information. On fera donc l'hypothèse suivante :

H est indiscernable d'une matrice aléatoire

C.3 Algorithmes par ensemble d'information

Dans toute la suite on considèrera une instance du problème 2 : $H \in \mathbb{F}_2^{r\times n}$, $s \in \mathbb{F}_2^r$ et $w \in \mathbb{N}^*$. On supposera que $w \leq t_{GV}$ de façon à s'assurer qu'il n'y ait qu'une solution. De plus on notera $H = (h_{i,j})_{(i,j)\in\{1,\cdots,r\}\times\{1,\cdots,n\}}$.

C.3.1 L'algorithme de Prange

Dans cette sous-section nous allons présenter l'algorithme de Prange. Nous commençons par une discussion informelle, s'ensuivra une description plus rigoureuse avec un calcul de complexité.

Étant donné que dans le système de Niederreiter le poids de l'erreur est connu, les solutions de $CSD(H, w, s)$ que l'on cherchera à calculer seront de poids exactement w. Résoudre $CSD(H, w, s)$ est équivalent à trouver w colonnes de H : $h_{j_1}, \cdots, h_{j_w}$ telles que

$$\sum_{i=1}^{w} h_{j_i} = s$$

Initialement on cherche à résoudre en les e_i le système :

$$\begin{cases} s_1 &= h_{1,1}\cdot e_1 + h_{1,2}\cdot e_2 + \cdots + h_{1,n}\cdot e_n \\ &\vdots \\ s_r &= h_{r,1}\cdot e_1 + h_{n,2}\cdot e_2 + \cdots + h_{r,n}\cdot e_n \end{cases}$$

On dispose donc de r équations à n inconnues. L'idée de Prange consiste alors à tirer au hasard un ensemble de $n - r = k$ positions J où $\{1, \cdots, n\} - J = \{i_1, \cdots, i_r\}$ et supposer que $e_i = 0$ pour les $i \in J$. Le système devient alors :

$$\begin{cases} s_1 &= h_{1,i_1}\cdot e_{i_1} + h_{1,i_2}\cdot e_{i_2} + \cdots + h_{1,i_r}\cdot e_{i_r} \\ &\vdots \\ s_r &= h_{r,i_1}\cdot e_{i_1} + h_{n,i_2}\cdot e_{i_2} + \cdots + h_{r,i_r}\cdot e_{i_r} \end{cases}$$

On a alors un système à r équations et r inconnues. D'après la discussion sur les codes aléatoires un tel système est inversible avec une forte probabilité. On peut donc supposer ici que l'on peut le résoudre. On obtient alors un vecteur $(e_{i_j})_{1\leq j\leq r}$ que l'on complète sur les coordonnées manquantes par des 0. Le vecteur $(e_i)_{1\leq i\leq n}$ est alors une solution de $eH^t = s$. Il ne reste maintenant plus qu'à vérifier que ce vecteur est de poids $\leq w$. Si ce n'est pas le cas on recommence avec un autre ensemble J.

Remarque 16. *L'ensemble J que l'on tire dans l'algorithme de Prange est généralement appelé un ensemble d'information.*

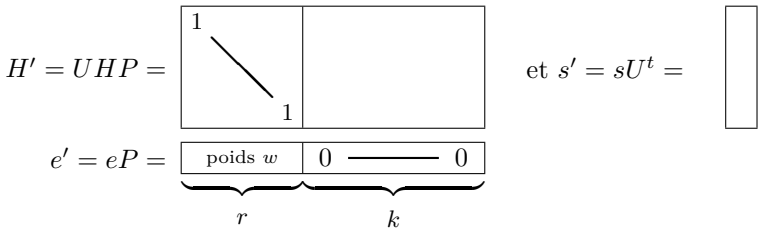
Nous pouvons ici adopter un point de vu de type code correcteur. Ce paradigme permet d'expliquer tout aussi simplement l'idée de Prange. Dans la première étape on a tiré un ensemble J sur lequel on va supposer qu'il n'y a pas d'erreurs. On considère ensuite le code correcteur poinçonné sur ces positions (on a supprimé des colonnes dans la matrice de parité) où cette fois-ci on peut décoder. On obtient alors un vecteur erreur mais ce dernier ne sera pas automatiquement une solution car il dépend de l'hypothèse que l'on a fait sur J. En effet, d'après la discussion qui précède, la solution à $CSD(H, s, w)$ est unique, donc le vecteur que l'on obtient sera solution si on a choisi un ensemble J sur lequel il n'y avait réellement pas d'erreurs. Toute la complexité de l'algorithme de Prange se résume donc à la probabilité de tirer initialement le bon ensemble. En effet, les autres étapes sont négligeables car il est très peu probable de tirer un ensemble J tel que $H_{\{1,\cdots,n\}-J}$ ne soit pas inversible et résoudre un système linéaire a un coût polynomial.

Étant donné que l'entrée de notre algorithme est une matrice, il reste un dernier point de vue, tout aussi instructif, celui matriciel. Le complémentaire $K = \{1, \cdots, n\} - J$ de l'ensemble que l'on a tiré est tel que H_K est inversible. Matriciellement celle correspond à procéder de la façon suivante : on tire une matrice P de permutation des colonnes et on effectue un pivot de Gauss sur HP. Le tirage de la matrice P revient à

choisir un ensemble J comme précédemment. Il s'agit de l'ensemble d'indice de colonnes $\{j_1, \cdots, j_k\}$ tel que P envoie ces colonnes sur les indices $\{r+1, \cdots, n\}$. Ceci donne donc une matrice de la forme $UHP = [I_r|H_0]$. Dans ce cas résoudre $CSD(H, s, w)$ est équivalent à résoudre $CSD(UHP, sU^t, w)$:

$$\begin{aligned} eH^t = s &\iff eH^tU^t = sU^t \\ &\iff ePP^tH^tU^t = sU^t \end{aligned}$$

où passer d'une solution à l'autre revient à multiplier par P ou P^t ce qui ne modifie pas le poids de Hamming (cf permutation). Lors de notre choix d'ensemble J, si nous avons de la chance, les erreurs vont se retrouver dans le complémentaire. Dans ce cas il est facile de vérifier qu'une solution e' de $CSD(UHP, sU^t, w)$ est de la forme $(s'|0)$. Donc si $w_H(s') = w$ nous aurons résolu le problème. Nous résumons cela à la figure qui suit.



Nous sommes maintenant en mesure de présenter formellement l'algorithme de Prange :

Algorithme de Prange

Input : $H \in \mathbb{F}_2^{r\times n}$, $s \in \mathbb{F}_2^r$, $w \in \mathbb{N}^*$
Output : $e \in \mathbb{F}_2^n$ tel que $eH^t = s$ et $w_H(e) = w$
$Boole \leftarrow false$
while $(Boole = false)$ **do**
    On choisit une matrice de permutation $P \in \mathbb{F}_2^{n\times n}$ aléatoirement
    $(H', U) \leftarrow PGElim(HP)$ /* On fait un pivot de Gauss */
    $s' \leftarrow sU^t$
    if $w_H(s') = w$ **then**
        $Boole \leftarrow true$ et $e \leftarrow (s'|0)P^t$
    end if
end while

Dans chaque tour de boucle on fait appel à un pivot de Gauss qui a un coût polynomial. Comme nous l'avons expliqué dans le rapport, nous négligeons ici les facteurs polynomiaux. Donc d'après le lemme qui suit, la complexité moyenne de l'algorithme de Prange est donc donnée par l'inverse de la probabilité de trouver une bonne matrice P ou de façon équivalente un bon ensemble J.

Lemme 2. *Soit $(X_k)_{k\geq 1}$ une suite de variables aléatoires de Bernoulli indépendantes et de paramètre p. On note Y la variable aléatoire :*

$$Y = \inf\{k \in \mathbb{N},\ X_k = 1\}$$

Alors $\mathbb{E}(Y) = \frac{1}{p}$.

Démonstration. Les variables aléatoires étant indépendantes, il est clair que :

$$\mathbb{P}(Y = k) = (1-p)^{k-1} \cdot p$$

Donc

$$\mathbb{E}(Y) = p\sum_{k=1}^{+\infty} k \cdot (1-p)^{k-1}$$

Ce qui donne par dérivation d'une série entière la résultat. □

Proposition 11 (Complexité de l'algorithme de Prange)**.**
La complexité de l'algorithme de Prange est donnée par :

$$\tilde{O}\left(\frac{\binom{n}{w}}{\binom{r}{w}}\right)$$

Démonstration. Dans le modèle de cryptanalyse du système de chiffrement de Niederreiter on suppose que les vecteurs erreurs sont tirés aléatoirement et ceci de façon uniforme. La probabilité de tirer le bon ensemble J se fait donc sur les motifs d'erreurs. Cette dernière est alors donnée par le nombre d'erreur de poids w ayant un support inclue dans un ensemble de taille r divisé par le nombre d'erreurs de poids w :

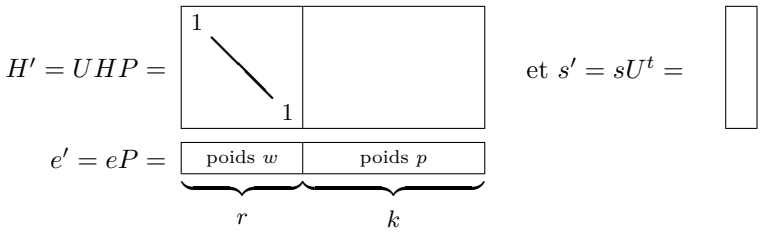
$$\frac{\binom{r}{w}}{\binom{n}{w}}$$

d'où le résultat.

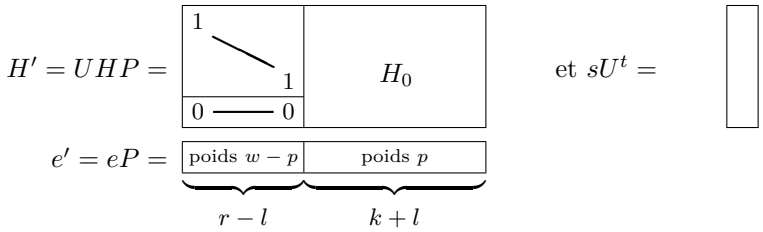
□

C.3.2 L'amélioration de Dumer

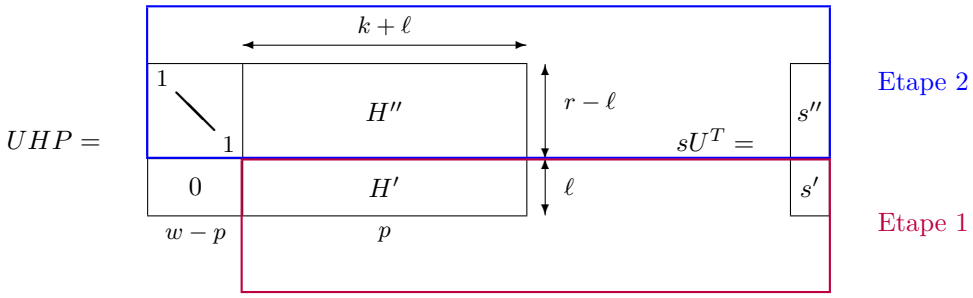
Comme nous venons de le voir, dans l'algorithme de Prange on cherche à tirer un ensemble J sur lequel il n'y a pas d'erreurs. Léon et Brickell ont alors proposé d'accepter qu'il y ait un petit nombre p d'erreurs sur cet ensemble J. On peut résumer cette idée avec la figure suivante :



De cette façon on a plus de chance de trouver le bon ensemble J. En revanche il va falloir énumérer tous les motifs d'erreurs de poids p sur la fenêtre de taille k. Il faudra donc trouver le paramètre p qui optimise l'algorithme. De façon un peu plus formelle : on choisit une matrice de permutation des colonnes P, on effectue un pivot de Gauss, on énumère $\{sU^t + e_0H_0^t \mid w_H(e_0) = p\}$ et si s' appartient à cette liste pour un vecteur e_0 et est de poids $w - p$ on renvoie le vecteur $(s'|e_0)P^t$. Le problème maintenant porte sur l'énumération. Cette étape peut être faite de façon naïve mais on peut faire mieux. La nouveauté de l'algorithme de Leon et Brickell est de chercher un mot e_0 de poids p tel $e_0H_0^t = 0$. L'idée de Dumer est de modifier légèrement ce problème pour le résoudre avec le paradoxe des anniversaires. Pour cela, ce dernier propose de commencer par faire un pivot de Gauss partiel où l'on « s'arrête » à I_{r-l} avec l un nouveau paramètre. On voit alors la matrice UHP comme :

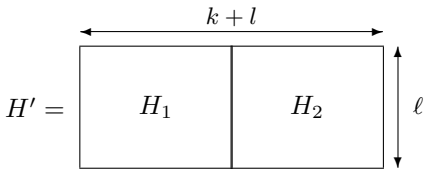


On sépare ensuite l'algorithme de Prange en deux étapes. On résume la situation avec la figure qui suit :



Dans l'étape 1 on va chercher l'ensemble des vecteurs $e_0 \in CSD(H', s', p)$, dans l'étape 2 on vérifie que $w_H(e'H''^t + s'') = w - p$ et on renvoie $(e'H''^t + s''|e')$.

Le problème maintenant est de résoudre $CSD(H', s', p)$. Pour cela, comme nous l'avons évoqué, Dumer utilise le paradoxe des anniversaires. Plus formellement, il propose de commencer par découper la matrice H' en deux :



On construit ensuite les deux listes :

$$\mathcal{L}_1 = \left\{ e_1 H_1^t \mid e_1 \in \mathbb{F}_2^{(k+l)/2} \text{ et } w_H(e_1) = \frac{p}{2} \right\} \quad ; \quad \mathcal{L}_2 = \left\{ s' + e_2 H_2^t \mid e_2 \in \mathbb{F}_2^{(k+l)/2} \text{ et } w_H(e_2) = \frac{p}{2} \right\}$$

On calcule leur intersection, c'est à dire l'ensemble des mots $e_1, e_2 \in \mathbb{F}_2^{(k+l)/2}$ de poids $p/2$ tels que $e_1 H_1^t = s' + e_2 H_2^t$. On renvoie ensuite le vecteur $(e_1|e_2)$ qui donne bien une solution de $CSD(H', s', p)$.

L'expression de paradoxe des anniversaires pour cette technique algorithmique a pour origine la proposition qui suit :

Proposition 12. *Soient L_1, L_2 deux listes de vecteurs de $\{0,1\}^l$ dont les entrées sont aléatoires et distribuées de façon uniforme. Alors l'espérance de la taille de leur intersection est :*

$$\frac{|L_1| \cdot |L_2|}{2^l}$$

Démonstration. On introduit la famille de variables aléatoires :

$$\forall (i,j) \in \{1, \cdots, |L_1|\} \times \{1, \cdots, |L_2|\},\ X_{i,j} = \begin{cases} 1 & \text{si } L_1[i] = L_2[j] \\ 0 & \text{sinon} \end{cases}$$

L'espérance que l'on cherche à calculer est donc :

$$\mathbb{E}\left(\sum_{(i,j) \in \{1, \cdots, |L_1|\} \times \{1, \cdots, |L_2|\}} X_{i,j} \right)$$

Les tirages se faisant sur deux listes où les éléments sont tirés de façon uniforme et indépendamment, $\mathbb{P}(X_{i,j} = 1)$ ne dépend pas du couple et est égale à $\frac{1}{2^l}$, d'où par linéarité de l'espérance :

$$\mathbb{E}\left(\sum_{(i,j) \in \{1, \cdots, |L_1|\} \times \{1, \cdots, |L_2|\}} X_{i,j} \right) = \frac{1}{2^l} |L_1| \cdot |L_2|$$

ce qui donne le résultat. □

Comme dans notre modèle de cryptanalyse, nous supposons que la matrice H est aléatoire, on peut appliquer la proposition qui précède et affirmer que les listes $\mathcal{L}_1$ et $\mathcal{L}_2$ auront en moyenne une intersection de taille :

$$\frac{|\mathcal{L}_1| \cdot |\mathcal{L}_2|}{2^l}$$

Ce résultat nous sera utile pour le calcul de la complexité de l'algorithme de Dumer que nous présentons formellement :

On note $PGElimDum$ l'algorithme qui prenant comme entrée HP renvoie les matrices U, H'', H_1 et H_2 que nous venons de définir.

Algorithme de Dumer

Input : $H \in \mathbb{F}_2^{r \times n}$, $w \in \mathbb{N}^*$, $l \in \{0, \cdots, r\}$ et $p \leq w$
Output : $e \in \mathbb{F}_2^n$ tel que $eH^t = s$ et $w_H(e) = w$
$Boole \leftarrow false$
while $(Boole = false)$ **do**
    On choisit une matrice de permutation $P \in \mathbb{F}_2^{n \times n}$ aléatoirement
    $(U, H'', H_1, H_2) \leftarrow PGElimDum(HP)$
    $s' \leftarrow sU^t$
    $\mathcal{S} \leftarrow [\,]$ /*liste vide*/
    $\mathcal{T} \leftarrow [\,]$ /*table de hachage*/
    for all $e_1 \in \mathbb{F}_2^{(k+l)/2}$ de poids $p/2$ **do**
        $x \leftarrow e_1 H_1^t$
        $\mathcal{T}[x] \leftarrow \mathcal{T}[x] \cup \{e_1\}$
    end for
    for all $e_2 \in \mathbb{F}_2^{(k+l)/2}$ de poids $p/2$ **do**
        $x \leftarrow s' + H_2 e_2^t$
        for all $e_1 \in \mathcal{T}[x]$ **do**
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{(e_1|e_2)\}$
        end for
    end for
    for all $e' \in \mathcal{S}$ **do**
        if $w_H(e'H''^t + s'') = w - p$ **then**
            $Bool \leftarrow true$
            return $(e'H''^t + s''|e')$
        end if
    end for
end while

Proposition 13 (Complexité de l'algorithme de Dumer)**.**
La complexité de l'algorithme de Dumer est donnée par :

$$\tilde{O}\left(\frac{\binom{n}{w}}{\binom{r-l}{w-p}} \cdot \left(\frac{1}{\sqrt{\binom{k+l}{p}}} + \frac{1}{2^l}\right)\right)$$

Démonstration. Dans l'algorithme de Dumer, le nombre de tours de boucle que l'on effectuera en moyenne est donné par l'inverse de la probabilité que l'erreur ait $w - p$ positions de son support dans un ensemble de taille $r - l$ et p positions dans le complémentaire, c'est à dire :

$$\frac{\binom{n}{w}}{\binom{r-l}{w-p} \cdot \binom{k+l}{p}}$$

A chaque tour de boucle, on fait un pivot de Gauss dont la complexité rentre dans le facteur polynomial. Il nous faut maintenant commencer par estimer la complexité de la technique de paradoxe des anniversaires.

On commence par construire deux listes de taille $\binom{(k+l/2)}{p/2}$. Ces dernières ne demandant que des calculs matriciels, ceci a un coût de :

$$\tilde{O}\left(\binom{(k+l/2)}{p/2}\right)$$

Du fait de l'utilisation d'une table de hachage, le complexité en moyenne restante est la taille de l'intersection des deux listes en moyenne, c'est à dire :

$$\frac{\binom{(k+l/2)}{p/2}^2}{2^l}$$

Sur tous ces vecteurs on applique ensuite une opération matricielle et une vérification de poids, le coût sera donc :

$$\tilde{O}\left(\frac{\binom{(k+l/2)}{p/2}^2}{2^l}\right)$$

En utilisant maintenant la fait que :

$$\binom{(k+l/2)}{p/2}^2 = \tilde{O}\left(\binom{(k+l)}{p}\right)$$

on a le résultat.

□

Cette complexité dépend des deux nouveaux paramètres introduits p et l. Il faut donc chercher à la minimiser. Pour cela, on remarque expérimentalement que dans la plupart des cas cette complexité est minimale pour $l = \log_2(\sqrt{\binom{k+l}{p}})$. La complexité *optimale* de l'algorithme de Dumer est alors donnée par :

$$\tilde{O}\left(\min_{0\leq p\leq w} \frac{\binom{n}{w}}{\binom{r-l}{w-p}\cdot\sqrt{\binom{k+l}{p}}}\right)$$

Nous renvoyons à la figure C.1 pour constater l'amélioration asymptotique qu'apporte Dumer à l'algorithme de Prange dans le cas d'un régime d'erreurs sur la borne de Gilbert-Varshamov.

C.3.3 Le décodage par ensemble d'information généralisé

Comme nous venons de le voir Leon, Brickell et Dumer ont pour le problème du décodage générique pris la brique de base de l'algorithme de Prange et l'ont amélioré. May, Meurer et Thomae puis Becker, Joux, May et Meurer ont ensuite proposé successivement deux améliorations dans [**?**] et [**?**] (que l'on abrège par MMT et BJMM), à l'algorithme de Dumer. Dans la partie principale du rapport nous avons présenté en détail ces deux améliorations s'inspirant de la technique des représentations. Nous ne les reprendrons pas ici. En revanche, nous allons présenter le décodage par ensemble d'information généralisé qui a été introduit pour la première fois par M.Finiasz et N.Sendrier dans [**?**]. Ces derniers ont remarqué que toutes les améliorations faites à l'algorithme de Prange vérifient le même « schéma ». C'est d'ailleurs pour cette raison que nous rangeons tous ces algorithmes dans la même classe dite par ensemble d'information.

Pour $x \in \mathbb{F}_2^r$, on notera $h_l(x)$ (resp. $f_r(x)$) ses l derniers (resp. premiers) bits. On notera $PGelim_l(H_0P)$ l'algorithme de pivot de Gauss renvoyant une matrice H_0 et U inversible de taille r telles que :

$$UHP = \begin{array}{|c|c|} \hline \begin{array}{c} 1 \\ \diagdown \\ 1 \end{array} & \multirow{2}{*}{H_0} \\ \hline 0 \text{ —— } 0 & \\ \hline \end{array}$$

$r-l$ $k+l$

De plus, on notera W_1 et W_2 deux sous ensembles de $W_{k+l,p/2} = \{e \in \mathbb{F}_2^{k+l},\ w_H(e) = \frac{p}{2}\}$. L'algorithme d'ensemble d'information possède alors toutes ces entrées, en plus de l'instance de $CSD(H, s, w)$, qu'il faudra optimiser.

ISD généralisé

Input : $H \in \mathbb{F}_2^{r\times n}$, $s \in \mathbb{F}_2^r$, $w \in \mathbb{N}^*$, $l \in \{0, \cdots, r\}$, W_1 et W_2
Output : $e \in \mathbb{F}_2^n$ tel que $eH^t = s$ et $w_H(e) = w$
$Boole \leftarrow false$
while $(Boole = false)$ **do**
    On choisit une matrice de permutation $P \in \mathbb{F}_2^{n\times n}$ aléatoirement
    $(H_0, U) \leftarrow PGElim_l(HP)$ /* On fait un pivot de Gauss */
    $s' \leftarrow sU^t$
    for all $e_1 \in W_1$ **do**
        $i \leftarrow h_l(e_1 H_0^t)$
        $write(e_1, i)$ /* On range e dans une structure (comme une table de hachage) à l'indice i */
    end for
    for all $e_2 \in W_2$ **do**
        $i \leftarrow h_l(s' + e_2 H_0^t)$
        $S \leftarrow read(i)$ /* On extraie l'élément 1 stocké à l'indice i */
        for all $e_1 \in S$ **do**
            if $w_H(s' + (e_1 + e_2)H_0^t) = w - p$ **then**
                $Bool \leftarrow True$
                return $(e'|e_1 + e_2)$ où $e' = f_{r-l}(s' + (e_1 + e_2)H_0^t)$
            end if
        end for
    end for
end while

Pour $l = 0$ et $p = 0$ on retrouve l'algorithme de Prange. Tandis que pour $l, p > 0$ et $W_1 = W_2 = W_{k+l,p/2}$ on obtient l'algorithme de Dumer. L'amélioration de Dumer a été dans le calcul de W_1 et W_2. Les algorithmes MMT et BJMM proposent alors d'améliorer encore ces calculs en utilisant la technique des représentations. Nous renvoyons aux sections 3.3 et 3.4 pour plus de détails.

Un algorithme de décodage générique est dit par ensemble d'information (ISD) s'il vérifie la structure qui précède.

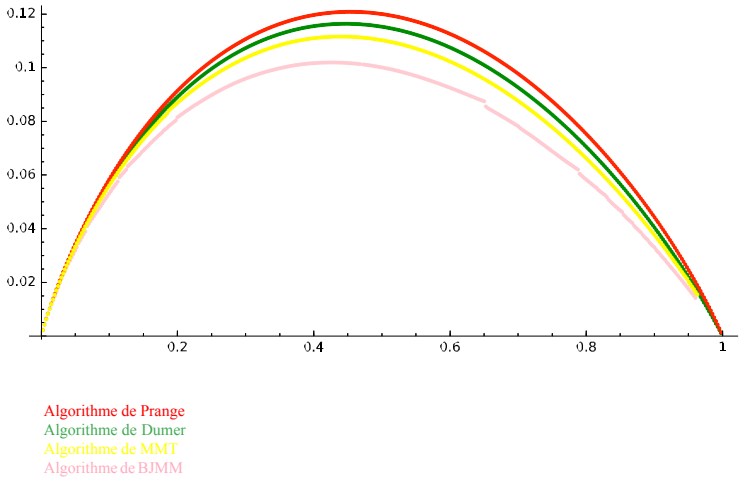
Définition 12 (Complexité de l'ISD)**.**
Pour toute valeur n, r et w, on notera $WF_{ISD}(n, r, w)$ l'exposant minimal en base 2 de l'algorithme ISD sur tous les choix de l, p, W_1 et W_2.

Comme nous l'avons souligné dans la sous-section qui précède, on mesure l'efficacité d'un algorithme de décodage générique pour un régime d'erreur sur la borne de Gilbert-Varshamov. Nous allons donc nous intéresser à $WF_{ISD}(n, r, w)$ dans le cas où $w = n \cdot h^{-1}(1 - R)$. On définit l'exposant asymptotique de l'algorithme de ISD comme :

$$C(R) = \lim_{n\to+\infty} \frac{1}{n} WF_{ISD}(n, n(1 - R), nh^{-1}(1 - R))$$

La figure qui suit vous donne les exposants asymptotiques, comme fonction de R, pour les algorithmes de Prange, Dumer, MMT et BJMM.

FIGURE C.1 – Comparaison des exposants asymptotiques des ISD



Annexe D

Théorie des types discrets

L'objectif de cette annexe est de présenter la théorie des types (discipline statistique s'appuyant sur la théorie de l'information). Nous nous sommes fortement inspirés du chapitre de [**?**].

Rappel : On appelle distance de Kullback pour deux probabilité de message p et q sur un alphabet $\mathcal{H}$,

$$D(p|q) = \sum_{x \in \mathcal{H}} p(x) \cdot \log_2 \frac{p(x)}{q(x)}$$

De plus on désignera par H la fonction entropie binaire.

D.1 Méthode des types

La méthode des types que nous allons présenter est un mélange des théories de l'information et des probabilités. Il s'agit d'une technique puissante qui va nous permettre de discerner des distributions. L'idée de cette méthode est « d'unifier » tous les échantillons ayant le même nombre d'occurrences (on ne tient plus compte de l'ordre d'apparition) de chaque valeur. Ce point de vu nous permettra d'obtenir les bornes les plus serrées possibles sur la taille des échantillons nécessaires aux problèmes de décision.

Dans toute la suite on désignera par $\mathcal{H}$ un alphabet $\{a_1, \cdots, a_{|\mathcal{H}|}\}$. On utilisera la notation x^n pour désigner une séquence de n éléments de $\mathcal{H}$: $x_1, \cdots, x_n$. De plus on notera $X_1, \cdots, X_n$ une suite de n variables aléatoires à valeurs dans $\mathcal{H}$. De même il pourra arriver de noter cette séquence X^n. Intuitivement x^n modélise un échantillon de n éléments et X^n l'ensemble des échantillons sous une distribution donnée.

L'objectif de la méthode des type est de calculer la probabilité d'apparition d'une séquence x^n sous une distribution fixée. Par exemple on peut modéliser le résultat de 7 lancers de dés par la séquence $x^7 = 1146232$. On cherchera ensuite à calculer la probabilité d'apparition de cette suite selon, par exemple, que le dé est truqué ou non.

Définition 13 (Type)**.**
On appelle type d'une séquence x^n l'application $\mathbb{P}_{x^n} : \mathcal{H} \to \mathbb{R}$ telle que

$$\forall a \in \mathcal{H}, \quad \mathbb{P}_{x^n}(a) = \frac{N(a|x)}{n} \text{ où } N(a|x) = \#\{i \in \{1, \cdots, n\}, x_i = a\}$$

On note $\mathcal{P}_n$ l'ensemble des types de $\mathcal{H}$ sur des séquences de longueur n.

Le type d'une séquence compte le nombre d'occurrence des symboles sans tenir compte de leur ordre d'apparition.

Remarque 17. *L'application $\mathbb{P}_{x^n}$ est une probabilité de masse sur $\mathcal{H}$.*

Définition 14 (Classe de type)**.**
Si $P \in \mathcal{P}_n$, on définit la classe de type de P comme :

$$T(P) = \{x^n \in \mathcal{H}^n,\ \mathbb{P}_{x^n} = P\}$$

La classe d'un type définit naturellement une relation d'équivalence sur les séquences. Considérer l'égalité modulo cette relation revient à ne plus considérer l'ordre d'apparition des éléments de $\mathcal{H}$.

Exemple 1. *Soient $\mathcal{H} = \{1, 2, 3\}$ et $x^5 = 11321$. Alors le type $\mathbb{P}_{x^5}$ est :*

$$\mathbb{P}_{x^5}(1) = \frac{3}{5}\ ;\ \mathbb{P}_{x^5}(2) = \frac{1}{5}\ ;\ \mathbb{P}_{x^5}(3) = \frac{1}{5}$$

La classe de type de $\mathbb{P}_{x^5}$ consiste en toutes les séquences de longueur 5 avec trois 1, un 0 et un 3. Il y a 20 telles séquences et,

$$T(\mathbb{P}_{x^5}) = \{11123, 11132, 11213, \cdots, 32111\}$$

Théorème 1. $|\mathcal{P}_n| \leq (n+1)^{|\mathcal{H}|}$

Démonstration. Un type est caractérisé par l'occurrence d'au plus n symboles de $\mathcal{H}$. Pour compter tous les types il suffit d'attribuer à chaque symbole son nombre d'occurrences possibles dans une séquence, c'est à dire $n+1$ (ne pas oublier la possibilité de 0 occurrence). On fait ceci sur tous les symboles et on obtient l'inégalité voulue. □

La remarque qui suit est le fondement de la théorie des types.

Remarque 18. *Le nombre d'échantillons possibles sur $\mathcal{H}$ étant $|\mathcal{H}|^n$, on pourrait s'attendre à ce qu'il y ait un nombre exponentiel de types de longueur n. Ce théorème affirme au contraire qu'il n'en existe qu'un nombre polynomial. Autrement dit un nombre exponentiel de séquences mène à la même distribution de probabilité.*

Si Q désigne une probabilité de masse sur $\mathcal{H}$, on désignera par Q^n la probabilité de masse sur $\mathcal{H}^n$ définie par :

$$Q^n(x^n) = \prod_{i=1}^{n} Q(x_i)$$

De même si $X_1, \cdots, X_n$ est une suite de variables aléatoires de distribution Q, on écrira

$$Q^n(x^n) = \prod_{i=1}^{n} Q(X_i = x_i)$$

Ces définitions se justifient par le fait que dans les échantillons considérés les tirages sont indépendants.

Théorème 2. *Soient $X_1, \cdots, X_n$ des v.a i.i.d de même distribution Q. Alors la probabilité d'une séquence ne dépend que de son type :*

$$Q^n(x^n) = 2^{-n(H(\mathbb{P}_{x^n}) + D(\mathbb{P}_{x^n}|Q))}$$

Démonstration.

$$\begin{aligned}
Q^n(x) &= \prod_{i=1}^{n} Q(x_i) \\
&= \prod_{a\in\mathcal{H}} Q(a)^{N(a|x)} \\
&= \prod_{a\in\mathcal{H}} Q(a)^{n\mathbb{P}_{x^n}(a)} \\
&= \prod_{a\in\mathcal{H}} 2^{\log_2(Q(a))\cdot n\mathbb{P}_{x^n}(a)} \\
&= \prod_{a\in\mathcal{H}} 2^{n\cdot(\mathbb{P}_{x^n}(a)\cdot\log_2 Q(a) - \mathbb{P}_{x^n}(a)\cdot\log_2(\mathbb{P}_{x^n}(a)) + \mathbb{P}_{x^n}(a)\cdot\log_2(\mathbb{P}_{x^n}(a)))} \\
&= 2^{n\cdot\sum_{a\in\mathcal{H}} -\mathbb{P}_{x^n}(a)\cdot\log_2\left(\frac{\mathbb{P}_{x^n}(a)}{Q(a)}\right) + \mathbb{P}_{x^n}(a)\cdot\log_2(\mathbb{P}_{x^n}(a))} \\
&= 2^{n\cdot(-D(\mathbb{P}_{x^n}|Q) - H(\mathbb{P}_{x^n}))}
\end{aligned}$$

□

On rappelle que $D(\mathbb{P}_{x^n}|Q)$ est en quelque la « distance » entre $\mathbb{P}_{x^n}$ et Q. Cette dernière quantifie l'erreur à supposer que la distribution est Q alors qu'en réalité il s'agit de $\mathbb{P}_{x^n}$. Le théorème 2 affirme donc que plus Q est « éloignée » de $\mathbb{P}_{x^n}$, plus la probabilité d'observer une séquence x^n sous la distribution Q est faible et ceci *de façon exponentielle*. Ce résultat est donc bien conforme à l'intuition.

Corollaire 1. *Si x^n est dans le classe de type de Q (je : $\mathbb{P}_{x^n} = Q$), alors :*

$$Q^n(x^n) = 2^{-nH(Q)}$$

Dire que x^n est dans la classe de type de Q revient à dire que $Q(x_i)$ est exactement le nombre d'apparitions de x_i dans x^n divisé par n.

Exemple 2. *D'après ce corollaire la probabilité qu'après n jets d'un dé non truqué on obtienne une séquence où il y a exactement $n/6$ occurrences de chaque symboles sera*

$$2^{-nH(\frac{1}{6},\cdots,\frac{1}{6})} = \frac{1}{6^n}$$

ce qui est évident. En revanche dans le cas d'un dé truqué sous la distribution $(\frac{1}{3},\frac{1}{3},\frac{1}{6},\frac{1}{12},\frac{1}{12},0)$ on obtient la formule

$$2^{-nH(\frac{1}{3},\frac{1}{3},\frac{1}{6},\frac{1}{12},\frac{1}{12},0)}$$

ce qui est bien plus clair que la formule obtenue de façon naturelle.

On va maintenant chercher à estimer le nombre de séquences menant à un même type, ce qui revient à donner des bornes sur $|T(P)|$.

Théorème 3 (Taille de $T(P)$)**.**
Pour tout type $P \in \mathcal{P}_n$,

$$\frac{1}{(n+1)^{|\mathcal{H}|}} 2^{nH(P)} \leq |T(P)| \leq 2^{nH(P)}$$

Démonstration. On peut calculer la taille exacte de $T(P)$ à partir des coefficients multinomiaux. Cependant cette valeur est difficile à manipuler, on ne donnera donc que des bornes sur la taille de $T(P)$.

Comme une probabilité est toujours inférieure à 1 on a :

$$\begin{aligned}
1 &\geq P^n(T(P)) \\
&= \sum_{x \in T(P)} P^n(x) \\
&= \sum_{x \in T(P)} 2^{-nH(P)} \text{ (cf corollaire (?))} \\
&= |T(P)| \cdot 2^{-nH(P)}
\end{aligned}$$

d'où la majoration. Démontrons maintenant la minoration. Pour cela on va démontrer que les séquences de $T(P)$ ont la probabilité d'apparition la plus élevée sous la distribution P, autrement dit

$$\forall \tilde{P} \in \mathcal{P}_n,\ P^n(T(\tilde{P})) \leq P^n(T(P))$$

En reprenant un calcul qui précède on a

$$\begin{aligned}
\frac{P^n(T(P))}{P^n(T(\tilde{P}))} &= \frac{|T(P)| \prod_{a \in \mathcal{H}} P(a)^{nP(a)}}{|T(\tilde{P})| \prod_{a \in \mathcal{H}} \tilde{P}(a)^{n\tilde{P}(a)}} \\
&= \frac{\binom{n}{nP(a_1), \cdots, nP(a_{|\mathcal{H}|})} \prod_{a \in \mathcal{H}} P(a)^{nP(a)}}{\binom{n}{n\tilde{P}(a_1), \cdots, n\tilde{P}(a_{|\mathcal{H}|})} \prod_{a \in \mathcal{H}} \tilde{P}(a)^{n\tilde{P}(a)}} \\
&= \prod_{a \in \mathcal{H}} \frac{(n\tilde{P}(a))!}{(nP(a))!} P(a)^{n(P(a)-\tilde{P}(a))}
\end{aligned}$$

En utilisant la minoration

$$\frac{m!}{n!} \geq n^{m-n}$$

on obtient

$$\begin{aligned}
\frac{P^n(T(P))}{P^n(T(\tilde{P}))} &\geq \prod_{a \in \mathcal{H}} (nP(a))^{n\tilde{P}(a)-nP(a)} (P(a))^{n(P(a)-\tilde{P}(a))} \\
&= \prod_{a \in \mathcal{H}} n^{n(\tilde{P}(a)-P(a))} \\
&= n^{n(\sum_{a \in \mathcal{H}} \tilde{P}(a) - \sum_{a \in \mathcal{H}} P(a))} \\
&= n^{n(1-1)} \\
&= 1
\end{aligned}$$

On obtient maintenant facilement la minoration du théorème,

$$\begin{aligned}
1 &= \sum_{Q\in\mathcal{P}_n} P^n(T(Q)) \\
&\leq \sum_{Q\in\mathcal{P}_n} \max_Q P^n(T(Q)) \\
&= \sum_{Q\in\mathcal{P}_n} P^n(T(P)) \\
&\leq (n+1)^{|\mathcal{H}|} P^n(T(P)) \\
&= (n+1)^{|\mathcal{H}|} \sum_{x\in T(P)} P^n(x) \\
&= (n+1)^{|\mathcal{H}|} \sum_{x\in T(P)} 2^{-nH(P)} \\
&= (n+1)^{|\mathcal{H}|} |T(P)| 2^{-nH(P)}
\end{aligned}$$

d'où le résultat. □

Exemple 3 (Application au cas binaire)**.**

Dans le cas où $\mathcal{H} = \{0,1\}$, un type est défini par le nombre de 1 apparaissant dans la séquence et dans ce cas la classe d'un type est de taille $\binom{n}{k}$. Le théorème montre donc que

$$\frac{1}{(n+1)^2} 2^{nH(\frac{k}{n})} \leq \binom{n}{k} \leq 2^{nH(\frac{k}{n})}$$

On peut cependant affiner la minoration en enlevant le carré du dénominateur.

Théorème 4 (Probabilité de classe de type)**.**

Pour tout type $P \in \mathcal{P}_n$ et toute distribution Q, la probabilité d'obtenir une séquence dans $T(P)$ sous la distribution Q est $2^{-nD(P|Q)}$ au premier ordre. Plus précisément,

$$\frac{1}{(n+1)^{|\mathcal{H}|}} 2^{-nD(P|Q)} \leq Q^n(T(P)) \leq 2^{-nD(P|Q)} \text{ où } Q^n(T(P)) = \sum_{x\in T(P)} Q^n(x)$$

Démonstration. D'après les résultats qui précèdent on a

$$\begin{aligned}
Q^n(T(P)) &= \sum_{x\in T(P)} Q^n(x) \\
&= \sum_{x\in T(P)} 2^{-n(D(P|Q)+H(P))} \\
&= |T(P)| \cdot 2^{-n(D(P|Q)+H(P))}
\end{aligned}$$

En utilisant les bornes sur $T(P)$ on obtient le résultat.

□

Ce théorème montre que si une distribution est « éloignée » de celle de la classe d'un type, alors la probabilité d'apparition d'une séquence de ce type est *exponentiellement faible.*

Nous pouvons résumer les théorèmes sur les types avec les quatres équations (où $\doteq$ est une égalité à un facteur polynomial près) :

$$|\mathcal{P}_n| \leq (n+1)^{|\mathcal{H}|} \tag{D.1}$$

$$Q^n(x) = 2^{-n(D(P_x|Q)+H(P_x))} \tag{D.2}$$

$$|T(P)| \doteq 2^{nH(P)} \tag{D.3}$$

$$Q^n(T(P)) \doteq 2^{-nD(P|Q)} \tag{D.4}$$

Avec ces équations nous pourront étudier le comportement d'échantillons sous une distribution fixée ce qui est l'objet de la sous-section qui suit.

D.2 Théorie des larges déviations

L'objectif de la théorie des larges déviations est de répondre à la question : **quelle est la probabilité qu'un système aléatoire dévie du comportement prédit par certaines lois des grands nombres ?**

Pour illustrer cette question nous allons commencer par un donner un exemple. On se donne n variables aléatoires, $X_1, \cdots, X_n$, indépendantes et de même loi $\mathcal{B}(1/3)$. Quelle est la probabilité que $(1/n)\sum_i X_i$ soit proche de $1/3$, proche de $3/4$? D'après la loi faible des grands nombres, la probabilité que $(1/n)\sum_i X_i$ soit proche de $1/3$ va tendre vers 1 avec n et 0 dans le cas de $3/4$. Il est légitime de se demander à quelle vitesse cette probabilité va tendre vers 0 ou 1 ? Autrement dit, à partir de quel ordre de grandeur sur n peut-on estimer que notre somme partielle est proche de son espérance ? Quelle est la probabilité en n d'être éloignée de l'espérance ? Le théorème central limite offre une réponse à ces questions. Malheureusement les bornes qu'il offre sur n sont bien trop larges. Lorsque nous chercherons à discerner deux distributions à l'aide d'échantillon nous chercherons au contraire à avoir les bornes les plus serrées possibles. En effet une contrainte naturelle du statisticien est de minimiser la taille des échantillons qu'il considère et ce qui lui permettra de prendre sa décision est un évènement « improbable » en fonction de cette taille. C'est la théorie des larges déviations qui va nous offrir des réponses « optimales ». Par exemple $(1/n)\sum X_i = \frac{3}{4}$ est équivalente à $\mathbb{P}_{x^n} = (1/4, 3/4)$ car ce type donne une proportion de 1 de $3/4$. De cette façon la probabilité d'obtenir $3/4$ est la probabilité d'obtenir ce type : $2^{-nD((3/4,1/4),(1/3,1/3))}$. Nous allons généraliser ces résultats à « un ensemble de possibilités ». Par exemple quelle est la probabilité d'observer $(1/n)\sum X_i \geq \frac{3}{4}$? Le but de cette sous-section est de formaliser « l'ensemble de possibilités » et de donner des bornes sur leurs réalisations. Nous verrons même que les bornes que nous donneront ne peuvent pas être améliorées.

Soit E un ensemble de probabilités de masse (intuitivement E représente l'ensemble des possibilités que nous venons d'évoquer). On note pour une distribution Q :

$$Q^n(E) = Q^n(E \cap \mathcal{P}^n) \quad = \sum_{x \,:\, P_x \in E \cap \mathcal{P}_n} Q^n(x)$$

$Q^n(E)$ est la probabilité de tirer des séquences sous la distribution Q qui correspondent à des types dans l'ensemble E. Dans la suite on confondra type d'une séquence et séquence.

Théorème 5 (Théorème de Sanov)**.**
Soient $X_1, \cdots, X_n$ des v.a i.i.d de même distribution Q. Soit E un ensemble de distributions de probabilité. Alors :

$$Q^n(E) \leq (n+1)^{|\mathcal{H}|} 2^{-nD(P^*|Q)}$$

où

$$P^* = \arg \inf_{P \in E} D(P|Q)$$

De plus si E est la fermeture de son intérieur, on a :

$$\frac{1}{n} \log_2 Q^n(E) \longrightarrow -D(P^*|Q)$$

La démonstration qui suit est importante car elle introduit un formalisme qui sera utile dans la suite. Ce formalisme permettra, entre autre, de comprendre avec rigueur dans quel espace nous nous plaçons pour faire des minimisations ou encore parler d'intérieur et de fermeture.

Démonstration. On identifie l'espace des probabilités de masse sur $\mathcal{H}$ à :

$$[0,1]^{|\mathcal{H}|}$$

que l'on munit de la métrique valeur absolue. On va commencer par démontrer la majoration en utilisant

les équations de la sous-section qui précède :

$$\begin{aligned} Q^n(E) &= \sum_{P\in E\cap\mathcal{P}_n} Q^n(T(P)) \\ &\leq \sum_{P\in E\cap\mathcal{P}_n} 2^{-nD(P|Q)} \\ &\leq \sum_{P\in E\cap\mathcal{P}_n} \max_{P\in E\cap\mathcal{P}_n} 2^{-nD(P|Q)} \\ &= \sum_{P\in E\cap\mathcal{P}_n} 2^{-n\min\limits_{P\in E\cap\mathcal{P}_n} D(P|Q)} \\ &\leq \sum_{P\in E\cap\mathcal{P}_n} 2^{-n\min\limits_{P\in E} D(P|Q)} \\ &= \sum_{P\in E\cap\mathcal{P}_n} 2^{-nD(P^*|Q)} \\ &\leq (n+1)^{|\mathcal{H}|}2^{-nD(P^*|Q)} \end{aligned}$$

Il faut noter ici que le minimum est une borne inférieure qui est atteinte en un élément de $[0,1]^{|\mathcal{H}|}$ car D est positive en ses paramètres.

Par définition de la borne inf il existe une suite $P_n \in E$ telle que : $D(P_n|Q) \underset{n\to+\infty}{\longrightarrow} D(P^*|Q)$.

On suppose maintenant que $\overline{\mathring{E}} = E$, de cette façon on peut supposer que la suite P_n introduite est dans $\mathring{E}$. Pour tout n_0 l'ensemble $\bigcup\limits_{n\geq n_0} \mathcal{P}_n$ est dense dans $[0,1]^{|\mathcal{H}|}$. Ce résultat se montre assez facilement mais étant uniquement technique nous l'omettons. On en déduit qu'il existe une suite Q_n telle que pour tout n, $Q_{(n)} \in \mathcal{P}_{(n)} \cap \mathring{E}$ telle que $|P_n - Q_n| \underset{n\to+\infty}{\longrightarrow} 0$ (dans l'espace métrique introduit). On montre alors facilement que :

$$D(Q_n|Q) \underset{n\to+\infty}{\longrightarrow} D(P^*|Q)$$

Pour tout n on a :

$$\begin{aligned} Q^n(E) &= \sum_{P\in E\cap\mathcal{P}_n} Q^n(T(P)) \\ &\geq Q^n(T(Q_n)) \\ &\geq \frac{1}{(n+1)^{|\mathcal{H}|}} 2^{-n\cdot D(Q_n|Q)} \end{aligned}$$

D'où

$$\begin{aligned} \liminf \frac{1}{n}\log_2 Q^n(E) &\geq \liminf \left(-\frac{|\mathcal{H}|\log_2(n+1)}{n} - D(Q_n|Q)\right) \\ &= -D(P^*|Q) \end{aligned}$$

Ceci donne une bonne inférieure sur la limite inf. Or avec ce qui précède la borne supérieure est identique, d'où le résultat de convergence annoncé. □

Remarque 19. *Le théorème de Sanov affirme que $Q^n(E)$ est inférieur à un facteur polynomial près à $2^{-nD(P^*|Q)}$. La deuxième affirmation du théorème nous donne encore plus d'informations : sous réserve d'hypothèses sur E, cette borne supérieure est optimale au niveau de l'exposant. On ne pourra donc faire mieux que d'affirmer :*

$$Q^n(E) \doteq 2^{-nD(P^*|Q)}$$

Exemple 4. *Ce théorème peut sembler obscur, on va donc donner un petit exemple d'application. On se donne fonction f définie sur $\mathcal{H}$ à valeurs dans $\mathbb{R}$. Considérons l'ensemble de distributions :*

$$\Pi = \{Q \ : \ \sum_{a\in\mathcal{H}} Q(a)f(a) \geq \alpha\}$$

Π est l'adhérence de son intérieur. Le type d'un échantillon $X_1, \cdots, X_n$ de loi Q appartient à cet ensemble si et seulement si $(1/n)\sum_i f(X_i) \geq \alpha$. Par le théorème de Sanov on a :

$$Q^n(\Pi) \dot{\leq} 2^{-nD(P^*|Q)} \text{ où } D(P^*|Q) = \min_{P\in\Pi} D(P|Q)$$

Pour trouver ce minimum on va utiliser le théorème des extrema liés sous les contraintes

$$\sum_{a\in\mathcal{H}} Q(a)f(a) \geq \alpha \text{ et } \{\sum_{a\in\mathcal{H}} Q(a) = 1\}$$

en considérant l'espace $[0,1]^{|\mathcal{H}|}$. Comme nous avons une inégalité dans la première contrainte on doit utiliser le théorème des extrema liés sur « chaque niveau ». Cependant nous allons démontrer que le minimum est atteint au niveau de α.

En différenciant $P \mapsto D(P|Q)$ et appliquant le théorème des extrema liés on a

$$\forall a \in \mathcal{H}, \ \log_2\left(\frac{P^*(x)}{Q(x)}\right) + 1 = \lambda f(x) + \nu$$

D'où

$$\forall x \in \mathcal{H}, \ P^*(x) = \frac{Q(x)e^{\lambda f(x)}}{\sum_{a\in\mathcal{H}} Q(a)e^{\lambda f(a)}}$$

On se donne maintenant $P \in \mathring{\Pi}$ et on va démontrer que $D(P|Q) > D(P^*|Q)$ ce qui donnera le résultat. On pose

$$c = \frac{1}{\sum_{a\in\mathcal{H}} Q(a)e^{\lambda f(a)}}$$

Quelques calculs donnent :

$$D(P^*|Q) = -\log_2(c) + \frac{\lambda}{\ln(2)}\alpha$$

$$\sum_{a\in\mathcal{H}} P(a)\log_2\left(\frac{P^*(a)}{Q(a)}\right) = \sum_{a\in\mathcal{H}} P(a)\cdot\left(-\log_2(c) + \frac{\lambda}{\ln(2)}f(a)\right) > -\log_2(c) + \frac{\lambda}{\ln(2)}\alpha$$

Ainsi

$$D(P|Q) - D(P^*|Q) > D(P|Q) - \sum_{a\in\mathcal{H}} P(a)\log_2\left(\frac{P^*(a)}{Q(a)}\right) = D(P|P^*) \geq 0$$

Donc P^* est bien un minimum.

D.3 Application aux tests statistiques d'hypothèses

L'un des principaux objectifs des statistiques est de décider entre deux hypothèses à partir de données observées. Par exemple, dans le domaine médical on aimerait savoir si un médicament est efficace ou non. Combien de fois doit-on le tester sur des patients avant de prendre une décision relevante ? Dans un jeu de pile ou face, combien de fois lancer une pièce avant de pouvoir « s'assurer » que celle-ci n'est pas truquée ?

Ces exemples illustrent un domaine d'étude des statistiques : les tests d'hypothèse. Ce problème général peut être illustré de la façon suivante :

Problème : Soient $X_1, \cdots, X_n$ des i.i.d de même loi Q. On considère les deux hypothèses :

1. $\mathcal{H}_1 : Q = P_1$;

2. $\mathcal{H}_2 : Q = P_2$;

Intuitivement $X_1, \cdots, X_n$ représente le résultat de n expériences ou dit autrement, un échantillon de taille n.

Se donner une « stratégie » pour prendre une décision entre $\mathcal{H}_1$ et $\mathcal{H}_2$ revient à se donner une fonction de décision $g(x_1, \cdots, x_n)$ sur les valeurs observées telle que si $g = 1$, on choisit $\mathcal{H}_1$ et si $g = 2$, on choisit $\mathcal{H}_2$. De cette façon on peut associer à notre test-stratégie une « région » A_n sur laquelle $g(x_1, \cdots, x_n)$ est égale à 1 et son complémentaire sur lequel $g(x_1, \cdots, x_n)$ vaut 2. Cependant il peut nous arriver de faire des erreurs, c'est à dire choisir $\mathcal{H}_1$ alors que nous sommes sous l'hypothèse $\mathcal{H}_2$ ou inversement. On définit donc les probabilités d'erreur relativement à la stratégie comme :

$$\alpha_n = P_1^n(A_n^c) \quad ; \quad \beta_n = P_2^n(A_n)$$

On peut alors résumer les différents cas de figure par le tableau :

Situation	conclusion	
	choix $\mathcal{H}_1$	choix $\mathcal{H}_2$
$\mathcal{H}_1$	pas d'erreur	erreur de type-I (fausse alerte)
$\mathcal{H}_2$	erreur de type-II (non-détectée)	pas d'erreur

α_n est la probabilité que sous la distribution de tirage P_1 on tombe dans A_n^c et donc que l'on fasse le choix de l'hypothèse $\mathcal{H}_2$ (inversement pour β_n).

En général on cherche à minimiser ces deux probabilités mais comme nous allons le voir il y a un compromis à faire.

Lemme 3 (Neyman-Pearson)**.**
En reprenant les notations qui précèdent, on choisit comme région de décision :

$$A_n(T) = \{(x_1, \cdots, x_n) \ : \ \frac{P_1(x_1, \cdots, x_n)}{P_2(x_1, \cdots, x_n)} > T\}$$

On note :

$$\alpha^* = P_1(A_n^c) \quad \textit{et} \quad \beta^* = P_2(A_n)$$

Alors pour toute autre région de décision (stratégie) de probabilité d'erreur α et β. Si $\alpha \leq \alpha^*$, alors $\beta \geq \beta^*$.

Il faut comprendre que la région que l'on a défini correspond à une stratégie. Si on observe $X_1 = x_1, \cdots, X_n = x_n$ et que $P_1(X_1 = x_1, \cdots, X_n = x_n) > T \cdot P_2(X_1 = x_1, \cdots, X_n = x_n)$ alors on choisit l'hypothèse 1. Dans le cas contraire on choisit l'hypothèse 2.

Remarque 20. *Le lemme de Neyman-Pearson indique que le test optimal est de la forme*

$$\frac{P_1(X_1, \cdots, X_n)}{P_2(X_1, \cdots, X_n)} > T$$

car aucune autre région ne pourra être meilleure sur les deux probabilités d'erreur. Dans le cas $T = 1$, on appelle ce test, le test de vraissemblance car on choisit l'évènement le plus probable.

On pose le ratio

$$L(X_1, \cdots, X_n) = \log_2\left(\frac{P_1(X_1, \cdots, X_n)}{P_2(X_1, \cdots, X_n)}\right)$$

qui est égale par indépendance des tirages d'échantillon à :

$$L(X_1, \cdots, X_n) = n\left(D(\mathbb{P}_{X^n}|P_2) - D(\mathbb{P}_{X^n}|P_1)\right)$$

De cette façon

$$\frac{P_1(X_1,\cdots,X_n)}{P_2(X_1,\cdots,X_n)} > T$$

est équivalent à

$$L(X_1,\cdots,X_n) = n\left(D(\mathbb{P}_{X^n}|P_2) - D(\mathbb{P}_{X^n}|P_1)\right) > \log_2(T)$$

Avec cette équivalence on peut considérer que le test est équivalent à spécifier une région de types B qui correspond à choisir l'hypothèse 1. Cette région de décision est ouverte et son complémentaire fermé. On peut donc appliquer le théorème de Savitch (on néglige le cas d'égalité) et les probabilités d'erreur s'écrivent :

$$\alpha_n = P_1(\mathbb{P}_{X^n} \in B^c) \doteq 2^{-nD(P_1^*|P_1)}$$

$$\beta_n = P_2(\mathbb{P}_{X^n} \in B) \doteq 2^{-nD(P_2^*|P_2)}$$

où P_1^* est l'élément le plus « proche » de P_1 dans $\overline{B^c}$ et P_2^* de P_2 dans B. Nous allons démontrer que

$$P_1^* = P_2^*$$

Minimisons $P \mapsto D(P|P_2)$ sous les contraintes $\{D(P|P_2) - D(P|P_1) \geq \frac{1}{n}\log_2(T)\}$ et $\{\sum_{a\in\mathcal{H}} P(a) = 1\}$. Tout comme dans l'exemple 4 nous allons montrer que le minimum se trouve sur la frontière $\frac{1}{n}\log_2(T)$. Soit P^* le minimum obtenue sous cette contrainte qui par le théorème des extrema liés s'écrit

$$\forall x \in \mathcal{H},\ P^*(x) = \frac{P_1^\lambda(x)P_2^{1-\lambda}(x)}{\sum_{a\in\mathcal{H}} P_1^\lambda(a)P_2^{1-\lambda}(a)}$$

On note c le dénominateur de cette expression. Quelques calculs donnent :

$$D(P^*|P_2) = -\log_2(c) + \lambda\cdot\frac{1}{n}\log_2(T)$$

La présence du facteur $\log_2(T)$ nous permet de conclure que P^* permet bien d'atteindre le minimum, d'où $P_2^* = P^*$.

La symétrie du raisonnement montre que la contrainte sur P_1^* se trouve aussi sur la frontière $\frac{1}{n}\log_2(T)$. De cette façon on obtient $P_1^* = P_2^*$ car le multiplicateur de Lagrange sera identique. En effet on a

$$D(P_1^*|P_1) = -\log_2(c) + (1-\mu)\cdot\frac{1}{n}\log_2(T)$$

et en égalisant

$$D(P_2^*|P_2) - D(P_1^*|P_1) = \frac{1}{n}\log_2(T)$$

on obtient $\lambda = \mu$. On note alors la distribution atteignant le minimum :

$$P_\lambda^*$$

Remarque 21. *Il faut bien comprendre que le coefficient λ dépend ici de la contrainte $\frac{1}{n}\log_2(T)$, c'est à dire du seuil T.*

Notre objectif maintenant va être de trouver le seuil T pour lequel la probabilité d'erreur est minimale. De plus, l'étude que nous avons fait du théorème de Sanov va nous permettre de montrer que l'exposant que nous trouverons ne pourra être meilleur : la borne de Chernoff.

Supposons que l'hypothèse $\mathcal{H}_1$ a une probabilité π_1 de se produire et π_2 pour $\mathcal{H}_2$. La probabilité d'erreur est alors :

$$P_e^{(n)} = \pi_1\alpha_n + \pi_2\beta_n$$

Soit

$$D^* = \lim_{n\to+\infty}\min_{A_n\subseteq\mathcal{H}^n} -\frac{1}{n}\log_2 P_e^{(n)}$$

Autrement dit D^* quantifie le meilleure exposant asymptotique pour toute région de décision. Le théorème qui suit nous sera d'une grande utilité dans l'étude du décodage statistique :

Théorème 6 (Borne de Chernoff)**.**
On a :

$$D^* = D(P_{\lambda^*}|P_1) = D(P_{\lambda^*}|P_2)$$

avec

$$P_\lambda = \frac{P_1^\lambda(x) \cdot P_2^{1-\lambda}(x)}{\sum_{a\in\mathcal{H}} P_1^\lambda(a) \cdot P_2^{1-\lambda}(a)}$$

et λ^* est la valeur de λ choisie de façon avoir égalité dans ce qui précède.

Démonstration. On reprend ici les notations et résultats qui précèdent. On en déduit que la probabilité d'erreur s'écrit :

$$P_e^{(n)} \doteq \pi_1 2^{-nD(P_\lambda^*|P_1)} + \pi_2 2^{-nD(P_\lambda^*|P_1)} \doteq 2^{-n\min(D(P_\lambda^*|P_1), D(P_\lambda^*|P_2))}$$

Comme nous l'avons souligné dans ce qui précède le λ dépend du seuil T (de la contrainte). Il nous faut ici trouver le λ « optimal ». Il est facile de vérifier que $D(P_\lambda^*|P_2)$ (resp. $D(P_\lambda^*|P_1)$) augmente (resp. diminue) avec λ. De cette façon le probabilité d'erreur est minimale lorsque $\min(D(P_\lambda^*|P_1), D(P_\lambda^*|P_2))$ est maximale, c'est à dire $D(P_\lambda^*|P_1) = D(P_\lambda^*|P_2)$. Or ce cas de figure est possible, il suffit de fixer le seuil T à 1.

Il est clair que cette probabilité d'erreur est optimale (on ne peut pas l'améliorer) d'après la discussion qui précède et l'utilisation du théorème de Savitch, d'où le résultat.

□

D.4 Une application : discerner deux variables de Bernoulli

On va maintenant appliquer tous les résultats qui précèdent au problème du discernement de deux variables de Bernoulli. Nous reprendrons les résultats de cette partie dans notre étude du décodage statistique.

Problème : Soient $X_1, \cdots, X_n$ des i.i.d de même loi Q. Les deux hypothèses que l'on considère sont
· $\mathcal{H}_0 : Q = P_1$ où $P_1 \sim \mathcal{B}(1/2 - \varepsilon)$;
· $\mathcal{H}_1 : Q = P_2$ où $P_2 \sim \mathcal{B}(1/2 + \varepsilon)$.

Un calcul simple montre que leur entropie relative est identique et égale à :

$$D(P_0|P_1) = D(P_1|P_0) = 2\varepsilon \cdot \log_2\left(1 + \frac{2\varepsilon}{1/2+\varepsilon}\right)$$

Asymptotiquement leur « distance va donc se comporter comme : $\sim 8\varepsilon^2/\ln(2)$.

Le lemme de Neymann-Pearson nous invite à considérer une région de décision de la forme :

$$A_n(T) = \left\{(x_1, \cdots, x_n) \ : \ D(\mathbb{P}_{X^n}|P_2) - D(\mathbb{P}_{X^n}|P_1) > \frac{1}{n}\log_2(T)\right\}$$

On note I_0 le nombre de 0 dans la séquence et I_1 le nombre de 1. Un calcul montre que cette région de décision est équivalente à :

$$(I_1 - I_0) \cdot \left(\log_2(\frac{1}{2} - \varepsilon) - \log_2(\frac{1}{2} + \varepsilon)\right) > \log_2(T)$$

D'après les résultats qui précèdent pour minimiser le probabilité d'erreur il faut prendre le seuil T à 1. La décision se prendra selon que les 1 (ou 0) sont majoritaires ou non. On va maintenant calculer la distribution qui minimise la probabilité d'erreur. Un calcul simple donne comme condition :

$$P_\lambda^*(0) = P_\lambda^*(1)$$

Donc $P_\lambda^* \sim \mathcal{B}(1/2)$. Le calcul de l'exposant de la borne de Chernoff donne :

$$\frac{1}{2}\left(-2 + \log_2\left(\frac{1}{2}-\varepsilon\right) + \log_2\left(\frac{1}{2}+\varepsilon\right)\right)$$

qui après un développement limité à l'ordre 3 donne :

$$\frac{2\varepsilon^2}{\ln(2)} + o(\varepsilon^3)$$

La théorie des types nous a permis de retrouver la proposition de Chernoff :

Proposition 14 (Borne de Chernoff)**.**

Soient $0 < p < 1$, $X_1, \cdots, X_k$ *des v.a i.i.d* $\sim \mathcal{B}(p)$ *et* $S_n = \sum_{k=0}^{n} X_k$. *Alors pour tout* $t \geq 0$ *:*

$$\mathbb{P}\left(|S_n - np| \geq n\varepsilon\right) \leq 2e^{-2n\varepsilon^2}$$

Cependant nous avons une information que ne donne pas cette proposition. L'exposant $-2n\varepsilon^2/ln(2)$ est optimal, on ne peut pas faire mieux. On en déduit qu'il faut prendre un échantillon de taille :

$$n \dot{\sim} \frac{1}{\varepsilon^2} \text{ (à un facteur polynomial en } \ln(\varepsilon) \text{ près)}$$

pour discerner deux variables de Bernoulli de paramètre distants de 2ε avec une probabilité d'erreur tendant vers 0. On peut alors résumer notre test de discernement par la figure D.1.

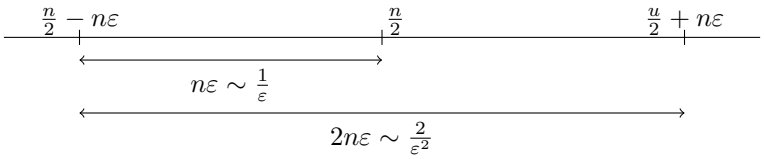


FIGURE D.1 – Decision region. Le côté gauche correspond à l'hypothèse $\mathcal{H}_0$ tandis que le côté droit correspond à $\mathcal{H}_1$.

Annexe E

Code Sage

Vous trouverez dans cette annexe le code Sage que j'ai réalisé afin de faire mes optimisations pour le poids des équations de parité du décodage statistique.

Nous commençons par le code permettant d'utiliser les fonctions entropie, entropie inverse et la borne de Gilbert-Varshamov comme fonction de R.

```
#The Shannon entropy function and its inverse are already
#implemented in a Sage library. We use these to define the
#binary entropy function.
#Reference : doc.sagemath.org/html/en/reference/coding/sage/coding/
#code_bounds.html
from sage.coding.code_bounds import *
from scipy.optimize import minimize

H = lambda x: float(entropy(x,2))
Hinv = lambda x: float(entropy_inverse(x,2))

def tracerEntropie(a):
milieu = 0.99
courbe = list()
for i in range(99):
milieu = milieu + 0.0001
courbe.append([milieu,Hinv(milieu)])
dess = list_plot(courbe,color='red')
dess.show()

#mindist est la fonction calculant la borne de Gilbert-Varshamov
#Reference : Anja Becker's thesis, appendix B

def mindist(R,e=0.00000001):
a = 0
b = 0.5
da = H(a)
db = H(b)
while (abs(da-db)>e):
testb = a + (b-a)/2
testdb = H(testb)
if (testdb> 1-R):
b = testb = a + (b-a)/2
```

```
db = H(b)
else:
a = a + (b-a)/2
da = H(a);
return a
```

Nous donnons maintenant le code pour l'algorithme de décodage Statistique avec l'amélioration de Dumer.

```
#Cette fonction retourne la valeur de pi qui minimise l'exposant du décodage
statistique avec l'amélioration de Dumer. On exprime ensuite l en fonction de pi.

def miniDumer(R,e=0.000000001):
a = 0
b = 0.1
milieu = 0
while (abs(b - a)>e):
milieu = a + (b-a)/2
if (milieu < -2*((R-milieu)/2 +
(1-R+milieu)*Hinv((2*milieu)/(1-R+milieu)))*log(1- 2*mindist(R),2)):
a = milieu
else:
b = milieu
if (milieu > R):
milieu = R
return milieu
```

Le code qui suit correspond à l'amélioration de MMT à un niveau.

```
def calculEpsilon(pi):
epsilon1 = ((1-pi) - sqrt(1-2*pi))/2
return epsilon1

def Expo(pi,l,R):
exp = (1-R+l)*H(((pi/2)+calculEpsilon(pi))/(1-R+l))
return exp

def calculr(R,pi,l):
if ((calculEpsilon(pi))/(1-R+l-pi) <=1):
if ((calculEpsilon(pi))/(1-R+l-pi) >=0):
r = pi + (1-R+l-pi)*H((calculEpsilon(pi))/(1-R+l-pi))
return r

def trouverlMMT(R,pi,e=0.0000000000000001):
a = 0
b = 1
milieu = 0
while (abs(b - a)>e):
milieu = a + (b-a)/2
exp = Expo(pi,milieu,R)
r = calculr(R,pi,milieu)
if (milieu < 2*exp - max(exp,r+(1/2)*exp)):
a = milieu
```

```
else:
b = milieu
return a


#Cette fonction retourne le pi du MMT décodage statistique

def minimiserMMT(R,e=0.0000000000000001):
a = 0
b = 0.1
milieu = 0
while (abs(b - a)>e):
milieu = a + (b-a)/2
l = trouverlMMT(R,milieu)
exp = Expo(milieu,l,R)
r = calculr(R,milieu,l)
if (2*Expo(milieu,l,R)-l-r<=((-2)*(milieu + ((R-l)/2))*log(1-2*mindist(R)))):
a = milieu
else:
b = milieu
return a
```

Cette portion de code donne les fonctions de l'amélioration de BJMM.

```
#BJMMdécodageStatistique

def maxi(a,b):
max = (1/2)*(abs(b-a) + a + b)
return max


#Cette fonction calcule le epsilon1 de BJMM comme fonction de pi

def calculEpsilon1(pi):
epsilon1 = ((1-pi) - sqrt(1-2*pi))/2
return epsilon1


#Cette fonction calcul le Epsilon2 de BJMM, c'est le premier qui est utilisé

def calculEpsilon2(pi):
epsilon2 = (1-(pi/2) - (pow(1-2*pi,(1/4))))/2
return epsilon2


#Cette fonction calcule le r2 de BJMM

def calculr2(pi,l,R,epsilon1,epsilon2):
r2 = (pi/2) + (1-R + l - (pi/2)-epsilon1)*H((epsilon2/(1-R+l - (pi/2)-epsilon1)))
return r2


#Cette fonction calcule le r2 de BJMM
```

```
def calculr1(pi,l,R,epsilon1):
r1 = pi + (1-R + l - pi)*H((epsilon1)/(1-R+l -pi))
return r1


#Cette fonction calcule l'exposant de la binomiale utilisée dans BJMM

def calculExp(pi,l,R,epsilon2):
exp = (1-R + l)*H(((pi/4) + epsilon2)/(1-R + l))
return exp


#Cette fonction permet de définir le l en fonction de pi
(résolution d'une équation définie avec un max)

def trouverlBJMM(R,pi,epsilon1,epsilon2,e=0.000000001):
a = 0
b = 1
milieu = 0
while (abs(b - a)>e):
milieu = a + (b-a)/2
expo = calculExp(pi,milieu,R,epsilon2)
if 4*expo - milieu -2*calculr2(pi,milieu,R,epsilon1,epsilon2)-
calculr1(pi,milieu,R,epsilon1) +
2*calculProbaDeuxNiveaux(R,pi,epsilon1,epsilon2,milieu) >
max(((1/2)*expo),expo - calculr2(pi,milieu,R,epsilon1,epsilon2),
2*expo- calculr1(pi,milieu,R,epsilon1)-calculr2(pi,milieu,R,epsilon1,epsilon2)):
a = milieu
else:
b = milieu
return a


#Cette fonction calcule le pi de BJMM en le choisissant de façon à ce que
la complexité de l'algorithme ne dépasse pas celle du décodage statistique
sans le phase de pré-calcul

def minimiserBJMM(R,epsilon1,epsilon2,e=0.0000001):
a = 0
b = 0.1
milieu = 0
while (abs(b - a)>e):
milieu = a + (b-a)/2
lo = trouverlBJMM(R,milieu,epsilon1,epsilon2)
expo = calculExp(milieu,lo,R,epsilon2)
if (4*expo-lo-2*calculr2(milieu,lo,R,epsilon1,epsilon2)
-calculr1(milieu,lo,R,epsilon1) +
2*calculProbaDeuxNiveaux(R,milieu,epsilon1,epsilon2,lo)  <
((-2)*(milieu + ((R-lo)/2))*log(1-2*mindist(R)))):
a = milieu
else:
b = milieu
return a
```

```
#Ces deux fonctions vont calculer les probabilités d'obtenir des mots
de bon poids dans BJMM à chaque niveau.

def calculProbaUnNiveau(R,pi,epsilon,l):
nu = ((pi/2)+epsilon)*H((epsilon)/((pi/2)+epsilon)) +
(1-R+l - (pi/2)-epsilon)*H((pi/2)/(1-R+l-(pi/2)-epsilon)) - exposant(pi,l,R,epsilon)
return nu


def calculProbaDeuxNiveaux(R,pi,epsilon1,epsilon2,l):
nuaux = ((pi/4)+epsilon2)*H((epsilon2 - (epsilon1/2))/((pi/4)+epsilon2)) +
(1-R+l - (pi/4)-epsilon2)*H(((pi/4)+(epsilon1/2))/(1-R+l-(pi/4)-epsilon2)) -
calculExp(pi,l,R,epsilon2)
return nuaux


def fonctionExposantBJMM(R,epsilon1,epsilon2):
pi = minimiserBJMM(R,epsilon1,epsilon2)
l = trouverlBJMM(R,pi,epsilon1,epsilon2)
nu1 = calculProbaUnNiveau(R,pi,epsilon1,l)
expos = decostat(pi + (1/2)*(R-l),R) - nu1
return expos


#Cette fonction permet de minimiser l'exposant du décodage statistique
qui est une fonction unimodale. On fixe epsilon2 pour trouver le minimum.

def GoldBJMM(R,epsilon2,e=0.00001):
gr = (math.sqrt(5) + 1) / 2
a = calculEpsilon1((minimiserMMT(R)))
b = 2*epsilon2
c = b -  (b - a) / gr
d = a + (b - a) / gr
while abs(c - d) > e:
if fonctionExposantBJMM(R,c,epsilon2) < fonctionExposantBJMM(R,d,epsilon2):
b = d
else:
a = c
c = b - (b - a) / gr
d = a + (b - a) / gr
return (b + a) / 2


#Cette fonction permet de trouver un epsilon2 de BJMM qui optimise l'exposant.

def GoldBJMM2(R,e=0.00001):
gr = (math.sqrt(5) + 1) / 2
a = calculEpsilon2((minimiserMMT(R)))
b = 20*a
c = b -  (b - a) / gr
d = a + (b - a) / gr
while abs(c - d) > e:
```

```
epsilon1c = GoldBJMM(R,c)
epsilon1d = GoldBJMM(R,d)
if fonctionExposantBJMM(R,epsilon1c,c) < fonctionExposantBJMM(R,epsilon1d,d):
b = d
else:
a = c
c = b - (b - a) / gr
d = a + (b - a) / gr
return (b + a) / 2


#Cette fonction revoie l'exposant du décodage statistique
avec l'utilisation de BJMM (après optimisation)

def fonctionExposantOptimalBJMM(R):
epsilon2 = GoldBJMM2(R)
epsilon1 = GoldBJMM(R,epsilon2)
pi = minimiserBJMM(R,epsilon1,epsilon2)
l = trouverlBJMM(R,pi,epsilon1,epsilon2)
nu1 = calculProbaUnNiveau(R,pi,epsilon1,l)
nu2 = calculProbaDeuxNiveaux(R,pi,epsilon1,epsilon2,l)
expos = fonctionExposantBJMM(R,epsilon1,epsilon2)
return expos
```

Annexe F

Quelques preuves

Proposition 15. *Soit $m \in \mathbb{F}_2^u$ et $m \neq 0$. Sur des tirages aléatoires uniformes de matrices $H \in \mathbb{F}_2^{u \times v}$, l'espérance de $w_H(mH)$ est $\frac{v}{2}$.*

Démonstration. Par hypothèse, les lignes de la matrices H sont constituées de v variables de Bernoulli de paramètre $1/2$. Or une somme modulo 2 d'un nombre quelconque de variables de Bernoulli de paramètre $1/2$ est encore une variable de Bernoulli de paramètre $1/2$, d'où le résultat. □

Proposition 16. *Supposons que $r \sim \log_2(N_{MMT})$. Alors pour toute solution e_s, une seule de ses représentations en moyenne sera dans les solutions finales.*

Démonstration. On va ici faire des probabilités sur la matrice G_2 en supposant que chacune de ses coordonnées est une variable de Bernoulli de paramètre $1/2$ (cf 1.2). On note $I^1, \cdots, I^{N_{MMT}}$ les représentations d'une même solution $e_1 + e_2$ où

$$I^i = (I_1^{(i)}, I_2^{(i)}, J^{(i)})$$

Soit $X_1, \cdots, X_{N_{MMT}}$ les v.a tels que $X_i(G_2) = 1$ si et seulement si

$$\sum_{j \in I_1^{(j)} \cup J^{(i)}} q_{j\,[r]} = 0_{[r]}$$

Une représentation d'une solution passera la première fusion si et seulement si elle vérifie l'égalité qui précède. La moyenne recherchée est donc l'espérance de $\sum_i X_i$. Or :

$$\mathbb{P}\left(\sum_{j \in I_1^{(j)} \cup J^{(i)}} q_{j\,[r]} = 0_{[r]} \right) = \frac{1}{2^r} \tag{F.1}$$

Les variables aléatoires étant indépendantes, on obtient alors :

$$\mathbb{E}\left(\sum_i X_i \right) = \frac{N_{MMT}}{2^r}$$

Donc si on choisit $r \sim \log_2(N_{MMT})$, on obtient une espérance égale à 1 et de cette façon une solution finale fixée n'admettra en moyenne qu'une représentation. Le paramètre r a bien joué son rôle de filtre.

□

Bibliographie

[1] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Comput. Sci.*, pages 1–16, Copenhagen, Denmark, May 2014. Springer.

[2] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In *Advances in Cryptology - EUROCRYPT 2012*, Lecture Notes in Comput. Sci. Springer, 2012.

[3] Elwyn Berlekamp, Robert McEliece, and Henk van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Inform. Theory*, 24(3) :384–386, May 1978.

[4] Rodolfo Canto-Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight, 2015. preprint.

[5] Thomas M. Cover and Joy A. Thomas. *Information Theory*. Wiley Series in Telecommunications. Wiley, 1991.

[6] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In M. Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Comput. Sci.*, pages 88–105. Springer, 2009.

[7] R. G. Gallager. *Low Density Parity Check Codes*. M.I.T. Press, Cambridge, Massachusetts, 1963.

[8] Abdulrahman Al Jabri. A statistical decoding algorithm for general linear block codes. In Bahram Honary, editor, *Cryptography and coding. Proceedings of the 8^{th} IMA International Conference*, volume 2260 of *Lecture Notes in Comput. Sci.*, pages 1–8, Cirencester, UK, December 2001. Springer.

[9] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $O(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Comput. Sci.*, pages 107–124. Springer, 2011.

[10] Raphael Overbeck. Statistical decoding revisited. In Reihaneh Safavi-Naini Lynn Batten, editor, *Information security and privacy : 11^{th} Australasian conference, ACISP 2006*, volume 4058 of *Lecture Notes in Comput. Sci.*, pages 283–294. Springer, 2006.

[11] Raphael Overbeck and Nicolas Sendrier. Code-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-quantum cryptography*, pages 95–145. Springer, 2009.

[12] Tom Richardson and Ruediger Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008.

[13] David Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Comput. Sci.*, pages 288–303. Springer, 2002.